國立交通大學

資訊科學與工程研究所

碩 士 論 文

DVB-RCS 網路中，FCA 機制對 TCP 效能的影響

The Effects of FCA Mechanism on TCP Performance over DVB-RCS Networks

研 究 生：邱耀宏

指導教授：王協源　教授

中 華 民 國　九 十 六　年 七 月

DVB-RCS 網路中，FCA 機制對 TCP 效能的影響
The Effects of FCA Mechanism on TCP Performance over DVB-RCS Networks

研 究 生：邱耀宏　　Student：Yao-Hung Chiu

指導教授：王協源　　Advisor：Shie-Yuan Wang

國 立 交 通 大 學

資 訊 科 學 與 工 程 系

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年七月

DVB-RCS 網路中，FCA 機制對 TCP 效能的影響

學生：邱耀宏　　指導教授：王協源 老師

國立交通大學資訊科學與工程學系（研究所）碩士班

摘　要

在 IP 網路盛行的今天，DVB-RCS 衛星網路越顯重要。許多的研究報告在探討 DVB-RCS 網路中的資源分配系統，不過鮮少提及其中的 FCA 機制，因此我們希望藉由模擬方式來探討 FCA 機制對 TCP 效能的影響。礙於目前尚無一套開發完整的 DVB-RCS 模擬平台可提供做為研究使用，我們在 NCTUns（交大網路模擬器）上開發一套 DVB-RCS 模擬平台。NCTUns 是一套開放的網路模擬器，我們所設計的 DVB-RCS 模擬平台將會在 NCTUns 4.0 釋出。除了探討 FCA 機制，本論文亦詳細描述 DVB-RCS 模擬平台的設計與實作，想要利用 NCTUns 作為研究工具的研究者可透過本論文認識該平台的核心架構。

The Effects of FCA Mechanism on TCP Performance over DVB-RCS Networks

Student：Yao-Hung Chiu　　Advisor：Dr. Shie-Yuan Wang

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

Many studies discuss the capacity allocation over DVB-RCS networks. However, few of them refer to the FCA mechanism. We discuss the effect of FCA mechanism on TCP performance over DVB-RCS networks via the simulation methodology. We develop a DVB-RCS simulation platform on NCTUns network simulator. In addition to the discussion over FCA mechanism, this thesis describes the design and implementation of the simulation platform.

誌　　謝

首先感謝恩師王協源教授多年以來對我的指導與照顧，除了讓我在課業上進步良多之外，也讓我學習到許多待人處事的道理。

感謝各位口試委員撥冗來到交通大學，聽取我們的論文報告並加以指導，有了委員們的建議，使得這篇論文得以更加完善。

感謝網路與系統實驗室的所有成員，有了你們這些伙伴的陪伴與支持，讓我在研究所兩年之中帶著愉快的心情學習。

最後，感謝父母對我的栽培，家人對我的照顧，有了他們的鼓勵，我才能夠完成本篇論文。

目　　錄

表　目　錄

圖　目　錄

一、 介紹

DVB-RCS [1] 是一套支援雙向溝通的衛星網路系統，DVB-RCS 以中央控管方式做資源管理，而 FCA 是 DVB-RCS 系統中負責分配資源的機制之一。自從 DVB-RCS 標準公開以來，許多的期刊論文及博士專題都在討論此標準，但其中鮮少對 FCA 機制做探討。

在 DVB-RCS 系統中有許多研究議題，比如服務品質（QoS, Quality of Service）、動態頻寬機制（BoD, Bandwidth on Demand）、以及網路效能等，研究此系統的研究者通常開發自己的模擬程式，用來評估提出的方案或驗證自己的分析 [2] [3]。這些模擬程式通常不是通用的而且也不是公開的；此外，許多模擬程式精確度並不是很高，使得模擬結果令人質疑。因此，一個公開且高品質的 DVB-RCS 網路模擬器是備受渴望的，它對於想開發、測試、或者改進 DVB-RCS 系統的研究者來說具有相當高的價值。

基於上述的動機，我們在交大網路模擬器（NCTUns Network Simulator） [4] 中設計並實作一個 DVB-RCS 網路模擬平台，系統中的機制（比如中央排程系統以及控制訊息交換等）皆受到嚴謹的實作、測試、以及驗證過程。我們在實作出的模擬平台上設計了許多模擬範例，在這些範例中觀察以及探討 FCA 機制對 TCP 效能的影響。

我們的貢獻主要有三項。首先，我們在通用的 NCTUns 模擬器上提供 DVB-RCS 網路模擬平台，除此之外，我們還在論文中呈現設計以及實作的細節，最後我們呈現 FCA 機制對 TCP 效能的影響。

接下來的章節，我們在第二章介紹 DVB-RCS 網路以及 NCTUns 網路模擬器，在第三章描述 DVB-RCS 網路模擬平台的設計與實作，接著在第四章對模擬平台做驗證，在第五章探討 FCA 機制對 TCP 效能的影響，最後在第六章做總結。

二、 背景

在衛星網路中，DVB-RCS 一直是備受討論的公開標準，在過去的十年中，數百篇的期刊論文及博士專題都在討論此標準。DVB-RCS 立基於 DVB 標準 [5] 制訂出一套可支援服務供應商與客戶端之間雙向溝通的系統，我們將在以下章節介紹 DVB 以及 DVB-RCS 的背景知識。爲了深入探討 DVB-RCS 系統，我們在 NCTUns 網路模擬器上開發 DVB-RCS 系統模擬功能，我們在 2.2 節對 NCTUns 網路模擬器做基本介紹。

2.1. 標準介紹

2.1.1. MPEG2 系統編碼

由於 DVB 標準採用 MPEG2 系統編碼 [6] 作爲傳輸格式，所以我們在介紹 DVB 標準前先稍微介紹 MPEG2 系統編碼。MPEG-2 是於 1994 年由 MPEG 工作團隊所發佈的視頻及音頻壓縮的國際標準。而 MPEG-2 通常用來爲廣播信號提供視頻及音頻編碼，其應用包括了數位衛星電視、有線電視等。MPEG-2 系統編碼並非是對影音編碼進行標準化，而是爲經過影音編碼的位元流以及其他未經過影音標碼的位元流提供了一種標準格式。MPEG-2 系統編碼的主要功能是將多條資料串流（整合前的串流稱爲「基本串流」）整合成單一條資料串流，使其適合儲存或傳輸介面。MPEG-2 系統編碼爲不同的需求提供兩組編碼格式：Transport Stream (TS)以及 Program Stream (PS)，TS 主要應用在不可靠的傳輸媒介，而 PS 主要應用在可靠的傳輸媒介上。

DVB 標準採用 MPEG-TS 格式，在 DVB 系統中，資料經由 MPEG2-TS 系統編碼後形成一個個封包，這些封包稱爲 MPEG2-TS 封包（MPEG2-TS Packet），而連續的 MPEG2-TS 封包被稱爲 MPEG2-TS 串流（MPEG2-TS Steam）。

2.1.2. DVB 標準

DVB 是由歐洲電信標準協會、歐洲電子標準化組織（CENELEC, European Committee for Electrotechnical Standardization）及歐洲廣播聯盟（EBU, European Broadcasting Union）聯合組成的「聯合專家組」（JTC, Joint Technical Committee）所發起的一系列國際承認的數位電視廣播標準，這套標準在歐洲使用的相當普及。DVB 提供服務供應商將資料傳送至客戶端的功能，以下我們將服務供應商至客戶端的傳輸通道稱爲正向通道（Forward Link）。

DVB 標準針對不同的需求提供了以下幾種資料廣播的模式：

- 資料管道（Data Piping）

- 資料流（Data Streaming）
- 多協定封裝（MPE, Multi-protocol Encapsulation）
- 資料傳送帶（Data Carousels）

資料管道：

資料管道模式提供簡單的、非同步的（Asynchronous）的資料傳送方式。資料管道模式並沒有定義封裝及切割的動作，所需廣播的資料作爲 MPEG2-TS 封包的負載（Payload）直接插入到 MPEG2-TS，如何由 MPEG2-TS 解析出資料是由應用層所負責。

資料流：

資料流模式定義了一種封裝方式，用以滿足同步（Synchronous）與同步化（Synchronized）資料串流的需求。同步資料串流適用在收端具有穩定播放速率的服務類型。爲了達到同步資料串需求，送端在送出的封包的標頭（Header）中置入時脈（Clock）資訊，可讓收端實現校時動作。同步化資料流適用在多個資料流之間必須達到同步回放（Play Back in Synchronization），比如一個服務可將電影的聲音以及影像資料分成兩個串流傳送，收端收到兩個串流可藉由標頭中的時脈資訊將聲音及影像同步回放。

多協定封裝：

多協定封裝提供了在 MPEG2-TS 之上支援通訊資料網路協定（Transporting Data Network Protocol）的一種機制。多協定封裝將通訊資料網路協定的協定資料單元（PDU, Protocol Data Unit）封裝成一種稱爲 MPE section 的封包格式。如果通訊資料網路協定的協定資料單元大於 4080 位元組（Byte），多協定封裝會將通訊資料網路協定的協定資料單元切割成多個 Section。多協定封裝採納 LLC／SNAP 協定來封裝其他協定的資料，而爲了對傳送 IP 協定做最佳化，允許在不接受 LLC／SNAP 封裝之下傳遞 IP 封包，但在這種封裝模式下，IP 封包的大小必須限制在 4080 位元組以下。

資料傳送帶：

資料傳送帶提供一套機制讓服務提供者循環式廣播一群資料模組。服務提供者可從資料傳送帶中及時的更新、添加或刪除內容；而接收端如果想要獲得特定資料模組中的內容，僅僅只需再等待該模再次被廣播就可以了。

在本論文中，我們將著重於「多協定封裝」的討論。這是因爲我們希望能夠觀察 IP 網路協定在這樣的系統下所得到的效能，而不是著重於視訊傳輸及音訊傳輸。

2.1.3. DVB-S 以及 DVB-S2 標準

DVB 系統傳輸方式有以下幾種：

- 衛星（DVB-S 及 DVB-S2）
- 有線電纜（DVB-C）

- 地面無線（DVB-T）
- 手持地面無線（DVB-H）

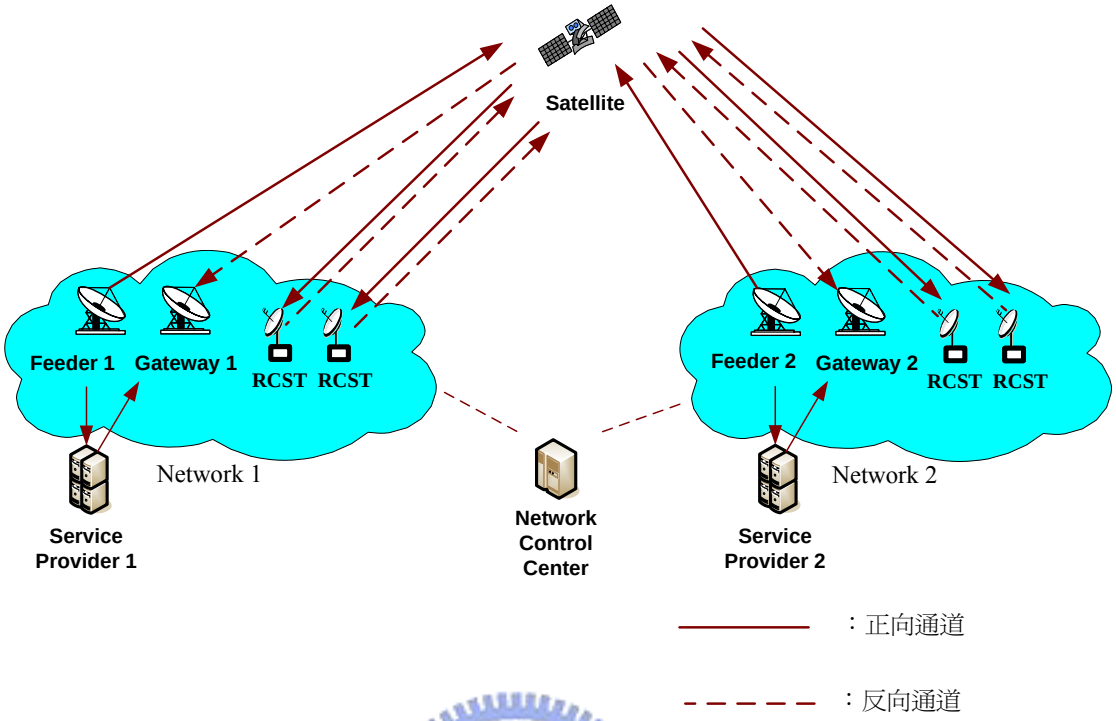


圖 1 DVB-RCS 系統架構圖

DVB-S 是使用空氣作爲傳輸媒介並使用衛星作爲轉送器（Transponder）的一套 DVB 系統。在近十年中，DVB-S 已經成爲最受喜好的數位電視廣播系統，尤其是在地域廣大的歐洲。用來取代 DVB-S 的新標準，稱爲 DVB-S2，已經在 2005 年制訂出。DVB-S2 標準引進了適應編碼調製（ACM, Adaptive Coding and Modulation），使得衛星系統的利用更爲有效率。

2.1.4. DVB-RCS 標準

在介紹完了 MPEG-2 及 DVB 後，我們可以發現 DVB 只有支援單向服務，也就是說資料只透過正向通道經由服務供應商到客戶端。而在這一節我們會引進反向通道（Return Link）的概念，也就是這節要介紹的 DVB-RCS（DVB-Return Channel via Satellite）標準。

圖 1 爲 ETSI 的 DVB-RCS 標準中互動式衛星網路的示意圖，我們列出在 DVB-RCS 中幾個重要的節點：

- 衛星（SAT, Satellite）
- 饋送者（FD, Feeder）
- 閘道（GW, Gateway）
- 網路控制中心（NCC, Network Central Control）

- 服務提供商（SP, Service Provider）
- 衛星地面接收站（RCST, Return Channel Satellite Terminal）

DVB-RCS 的網路系統中，所有送給客戶端的封包都是透過正向通道來傳送，反之由客戶端傳回的封包都是透過反向通道來傳送。一個 DVB-RCS 的網路系統是由一個或多個子網路所組成，每個子網路都有一個 FD 來將封包送入正向通道以及一個 GW 用來將封包送入反向通道，使這個子網路能利用衛星資源。整個 DVB-RCS 網路的資源都是由統一的 NCC 集中管理，使此網路中的每一個 RCST 可以分配到適當的網路資源。除了管理網路資源，NCC 亦負責控管 RCST 進出 DVB-RCS 網路。想要進入 DVB-RCS 網路的 RCST 向 NCC 發出登入請求，NCC 對登入請求有權決定是否允許。為了讓 DVB-RCS 網路可以與 Internet 網路互動，NCC 可以將其所擁有的資源下放給服務提供商，如此一來這些服務提供商就可以利用其原本 Internet 網路的資源及 DVB-RCS 網路中的 GW 和 FD 將 DVB-RCS 網路中的封包轉出到 Internet 網路或是將 Internet 網路中的封包轉入到 DVB-RCS 網路。RCST 是一個具有 IP 路由能力的客戶端設備，其後端可連接個人電腦或一個區域網路。

DVB-RCS 標準對於如何支援 IP 通訊協定有明確的定義。如圖 2 所示，IP 封包在正向通道的傳送有兩種封裝模式，其中一種是利用多協定封裝及 MPEG-2 TS，另一種是利用 ATM／AAL5 協定搭配 MPEG-2 TS，後者是用在新生代網路系統中提供 RCST 之間直接溝通，在本論文中我們僅討論第一種封裝模式。我們可看到 IP 封包在反向通道的傳送亦有兩種封裝模式，ATM／AAL5 協定或多協定封裝搭配 MPEG-2 TS，在本論文中我們對反向通道僅探討 ATM／AAL5 封裝模式。

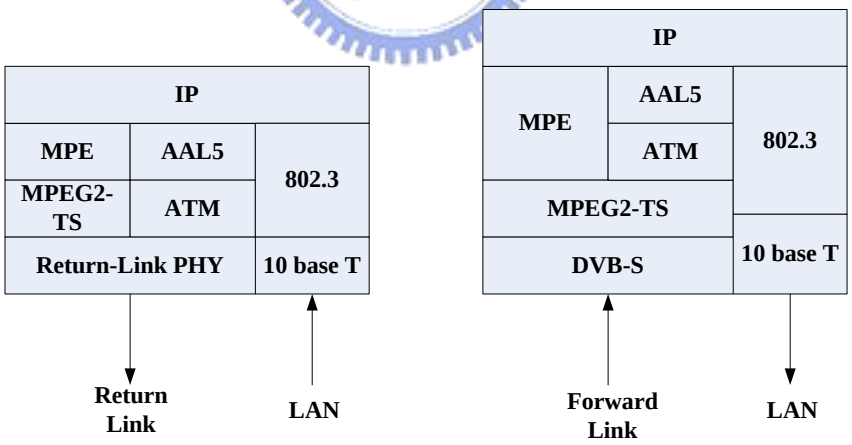


圖 2 正向通道及反向通道的協定堆

表 1 列出正向通道及反向通道在實體層（PHY, Physical Layer）所使用的調變（Modulation）及通道編碼（Channel Coding）、在媒體存取控制層（MAC, Media Access Control Layer）所使用的存取機制（Multiple Access Scheme）。

	正向通道	反向通道
調變模式	QPSK、8PSK、 16APSK、32APSK	QPSK
通道編碼	BCH code 搭配 LDPC code	Turbo code
		Convolution code 搭配 Reed-Solomon（RS）code
存取機制	TDM	MF-TDMA

表 1 正向通道及反向通道的比較

2.2. NCTUns 介紹

NCTUns 網路模擬器是一套具有高精準度以及可擴充性的網路模擬器。NCTUns 使用新穎的核心重入（Kernel Re-entering）模擬方法，此方法直接使用 UNIX 核心中的 TCP/IP 協定堆來產生模擬結果。同樣的，UNIX 上的應用程式也可以在不經過修改之下，直接執行在我們的模擬環境之中。

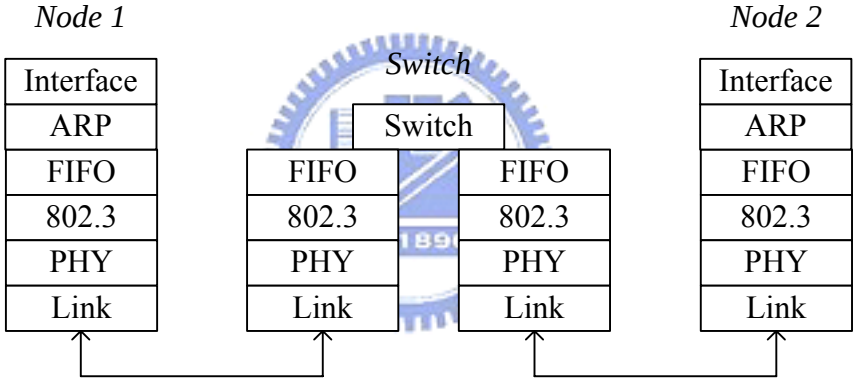
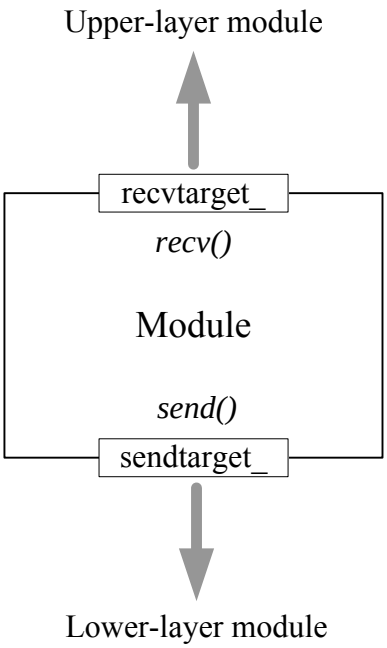


圖 3 模組化的網路架構

NCTUns 的模擬引擎使用分散式事件模擬方法（Discrete-event Simulation Method），因此，模擬的效能主要與事件個數相關，模擬之中產生越多的事件，則模擬速度就越慢。

NCTUns 採用開放式的系統架構，而且提供模組化的開發平台。圖 3 是模組化平台中一個網路拓樸範例，此拓樸由三個節點組成，各節點的架構也展現在圖中。圖 4 顯示出一個模組的骨幹，立基於此，研究者可以輕易地開發出他們自己的模組，並將開發出的模組整合至模擬器。




```
class NslObject
{
 private:
        char            *name_;
        u_int32_t       nodeID_;
        u_int32_t       portid_;
        u_int32_t       nodeType_;
 public:
        MBinder         *recvtarget_;
        MBinder         *sendtarget_;

        virtual inline int  int();
        virtual inline int  recv(ePacket_ *);
        virtual inline int  send(ePacket_ *);
        ...
};
```

圖 4 NCTUns 中的模組骨架

三、 DVB-RCS 模擬的設計與實作

我們在這一章展現我們在 NCTUns 網路模擬器上開發的 DVB-RCS 模擬平台。我們一開始先展示我們的 DVB-RCS 網路的概觀，隨後我們介紹 NCTUns 上爲了支援 DVB-RCS 網路而新增的模組，最後我們對 DVB-RCS 網路核心元件「通道資源分配」做詳細說明。

3.1. 網路拓樸

在 DVB-RCS 系統中我們設計了六種節點，如下圖 5。以 OSI 七層協定的角度可將此六種節區分爲三類，分別是：

- 第一層節點：SAT 節點、FD 節點和 GW 節點
- 第二層節點：NCC 節點
- 第三層節點：SP 節點以及 RCST 節點

第一層節點主要的工作是提供一個能與衛星溝通的界面，相距遙遠的兩個節點可以利用這些統一的界面透過衛星來溝通而不需要考慮與發送或接收衛星訊號的方法。

在我們的設計中只有一個節點爲第二層節點，即是 NCC 節點，它只負責 DVB-RCS 網路中所有的資源分配而不須要瞭解如何與外界的 Internet 網路溝通。

第三層節點分別是 SP 節點以及 RCST 節點，這兩種節點都有一個共通的工作，即它們都必須負責 DVB-RCS 網路與外界網路之間的溝通，所以這兩種節點上需要有路由（Routing）的能力，所以皆被設計爲第三層的節點。

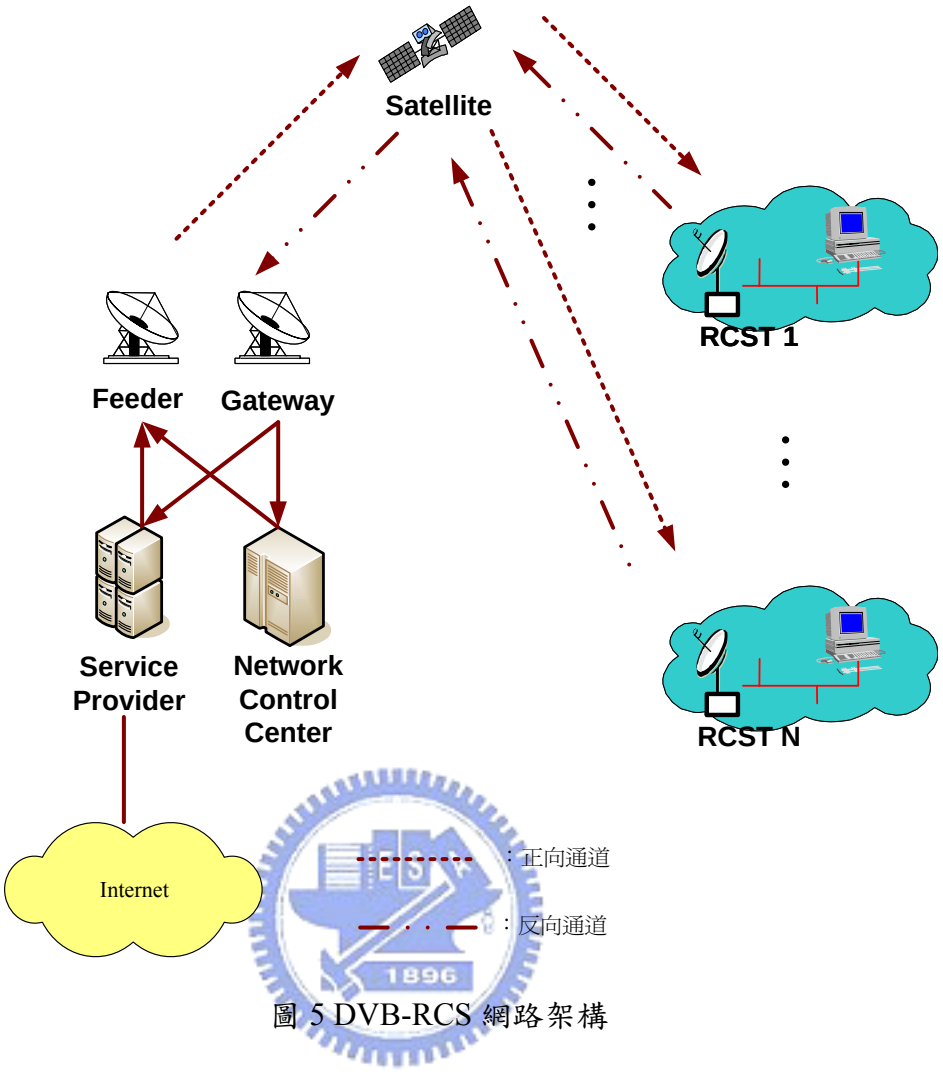


圖 5 DVB-RCS 網路架構

3.1.1. 正向通道上的資料流向

以下我們藉由圖 6 展現正向通道上資料的流向。任何要送至 RCST 的資料皆會先被路由至 SP 節點，如果資料的發送者爲 RCST 節點，資料必須先穿越反向通道到達 SP 節點。一旦 SP 節點收到資料並判斷該資料的接收者位址處於同屬的 DVB-RCS 網路，SP 節點把資料路由至 FD 節點。FD 節點將每筆收到的資料轉發至 SAT 節點，SAT 節點會將收到的資料廣播到 DVB-RCS 網路內各個 RCST 節點，各個 RCST 節點要負責判斷是否要接收正向通道送來的資料，最後 RCST 節點將接收到的資料路由至後端區域網路。

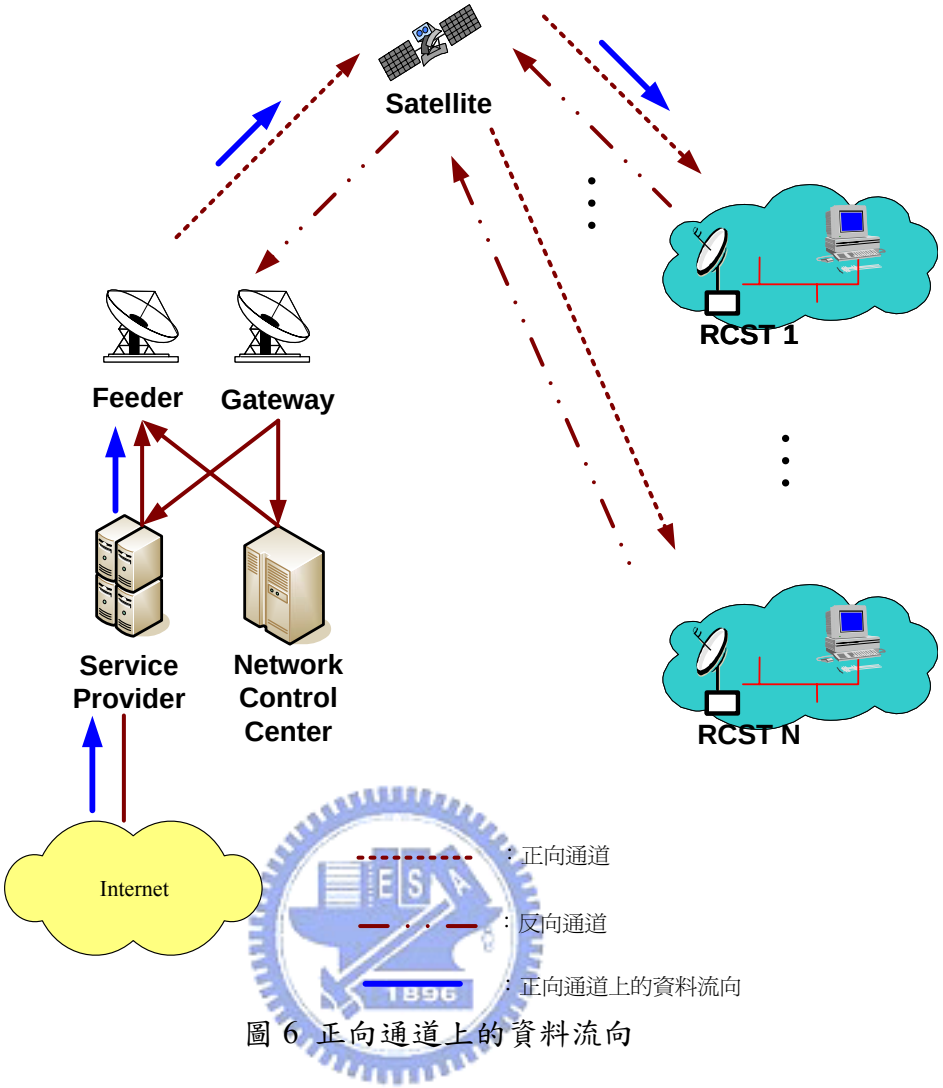


圖 6 正向通道上的資料流向

3.1.2. 反向通道上的資料流向

圖 7 展現反向通道上資料的流向。RCST 節點從後端區域網路接收到欲向外傳送的資料時，RCST 節點將收到的資料往 SAT 節點傳送。SAT 節點收到來自任一個 RCST 節點的資料，都會直接向 GW 節點轉發，而 GW 節點將 SAT 送來的資料送往 SP 節點。一旦 SP 節點收到資料，SP 視接收者位址決定把資料路由至 Internet 網路端或是將資料導入正向通道。

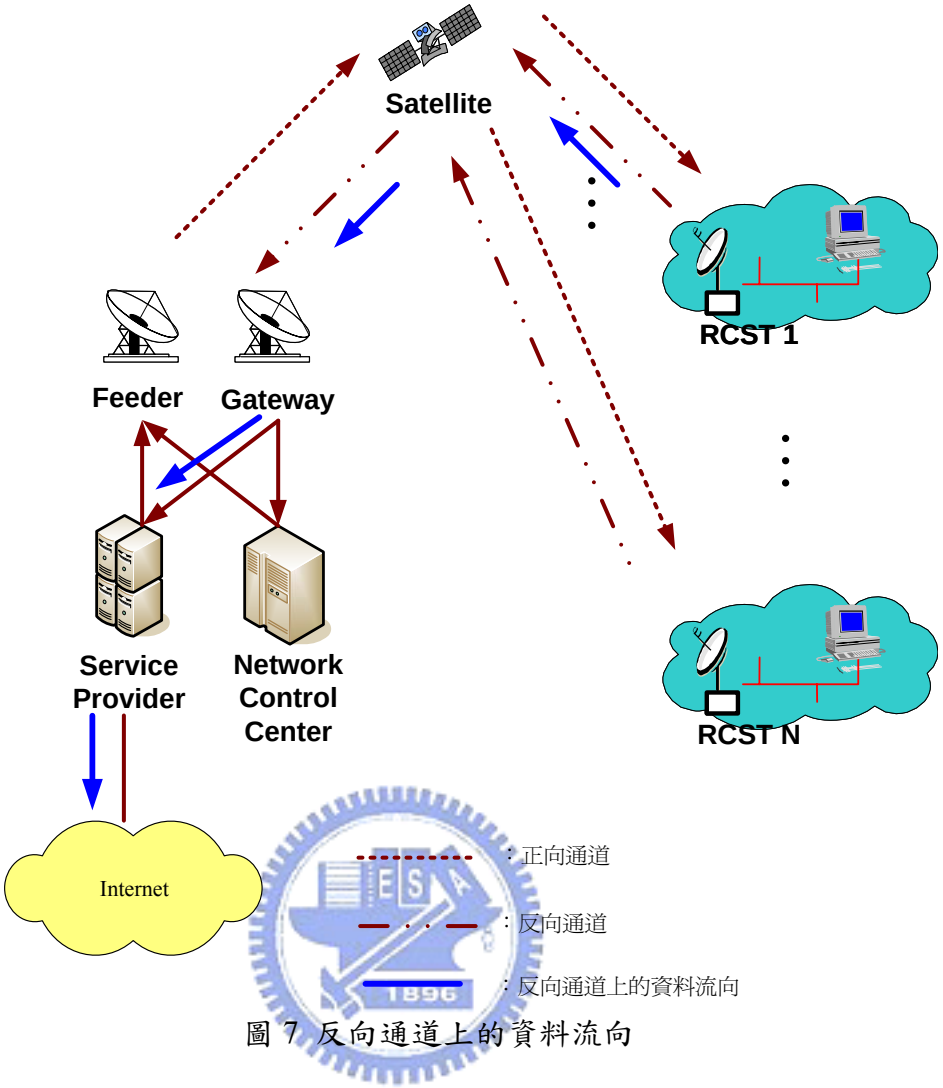


圖 7 反向通道上的資料流向

3.2. 控制訊息

3.2.1. 控制表格

爲了管理以及控制 DVB-RCS 網路系統，SP 節點以及 NCC 節點會透過正向通道廣播控制資訊到網路中所有 RCST 節點，這類控制訊息被稱爲「控制表格」(Control Table)。DVB-RCS 系統中控制表格有非常多的種類，但是有多種的控制表格是爲了提供廠商或使用者能夠更有彈性的管理或使用這個系統，因此在我們的設計之下，只有使用其中最重要的八種控制表格：PAT、PMT、NIT、INT、SCT、FCT、TCT、以及 TBTP。 以下我們概略介紹這八種控制表格的作用，控制表格詳細說明請參考 [7] [8]。

圖 8 正向通道上的多工/解多工

正向通道上的多工／解多工（Multiplexing/De-multiplexing）

在我們的系統中 PAT、PMT、NIT、以及 INT 四種控制表格用來達成正向通道上的多工／解多工。在第二章有提到 DVB-RCS 在正向通道上利用 MPEG2-TS 格式來傳輸資料，當 IP 封包要進入正向通道時，會被封裝成 MPE section 並切割成一或多個 MPEG2-TS 封包。而在我們的系統中，正向通道上送往一個 RCST 節點的資料被視爲一條基本串流，所有基本串流會被合併成一條 MPEG2-TS 串流以 TDM 方式送上正向通道，爲了做到多工／解多工每一條基本串流被給予一個唯一的辨認碼，稱爲 PID（Packet Identifier），也可視爲各個 RCST 擁有一個 PID 值。SP 節點會定期發送 PAT、PMT、NIT、以及 INT 控制表格，每一個 RCST 節點可透過查詢這四種表格找到屬於自己的 PID。以圖 8 爲例，Pa 表示一個送往 RCST A 的 IP 封包；Pb 表示一個送往 RCST B 的 IP 封包。在進入正向通道前 Pa 被切割成 Pa1、Pa2 兩個 MPEG2-TS 封包，每一個封包皆帶有 RCST A 的 PID 值 1；而 Pb 被切割成 Pb1、Pb2、Pb3 三個 MPEG2-TS 封包，每一個封包皆帶有 RCST B 的 PID 值 2。RCST A 知道自己的 PID 爲 1，所以當以上五個封包傳送至 RCST A 時，只會接收 Pa1 以及 Pa2 並組合回 IP 封包「Pa」；相同地，RCST B 接收 Pb1、Pb2、以及 Pb3 並組合回 IP 封包「Pb」。

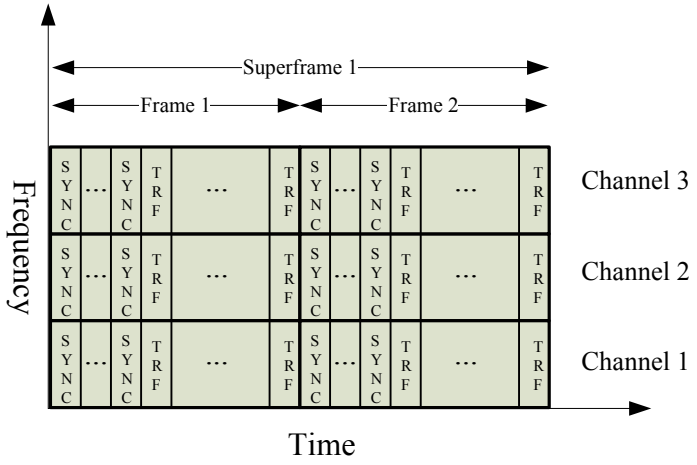


圖 9 反向通道上的頻道分割

反向通道的存取控制

在我們的系統中 SCT、FCT、TCT、以及 TBTP 四種控制表格用來達成反向通道的存取控制，介紹這四種控制表之前，我們先介紹反向通道的存取控制。圖 9 說明我們系統中反向通道架構，反向通道由一個個位於不同頻帶且頻率寬（Frequency Bandwidth）相同的頻道（Channel）組成，每一個 RCST 節點固定使用一個頻道，而使用相同頻道的所有 RCST 節點共同分享頻道資源。每一個頻道在時間上劃分成一個個相同長度的時段，稱爲超視框（Superframe）。每一個超視框被劃分成長度相同的時段，稱爲視框（Frame），視框是反向通道資源分配的週期，NCC 每間隔一個視框執行一次資源分配。每一個視框又被劃分成長度相同的時段，稱爲時槽（Timeslot），時槽是反向通道資源分配的基本單位，一個時槽可以傳送一個 Burst。我們的系統中實作了兩種 Burst：ATM traffic Burst 以及 SYNC（Synchronization） Burst。在 2.1.4 節中有提到 IP 封包在反向通道會被切割並封裝成 ATM 封包，而 ATM traffic Burst 便是用來傳遞 ATM 封包，也就是說 ATM traffic Burst 傳遞使用者資料。SYNC Burst 的主要用途是用來維持同步以及讓 RCST 節點回傳控制訊息，爲了簡化設計，我們在目前系統沒有模擬同步程序而僅模擬同步所需的額外負載（Overhead）。爲了讓 RCST 節點可傳遞要求訊息（Request Message）至 NCC 節點，DVB-RCS 標準制訂了四種傳遞方法，我們實作了其中一種，也就是將 RCST 的要求訊息裝載在 SYNC Burst。在以下章節我們把傳送 ATM traffic Burst 的時槽稱爲「資料時槽」（TRF Timeslot）；而把傳送 SYNC Burst 的時槽稱爲「要求時槽」（Request Timeslot）。

爲了讓網路中所有 RCST 節點知道如何利用反向通道，NCC 節點定期發送 SCT、FCT、TCT、以及 TBTP 四種控制表格。SCT 控制表格中主要描述各個超視框開始時間以及內部各個視框的相對位置；FCT 控制表格中主要描述視框內部各個時槽的相對位置；TCT 控制表格描述時槽的參數，比如符號傳輸率（Symbol Rate）、編碼率（Coding Rate）、負載類型等。藉由查詢 SCT、FCT、以及 TCT 三張控制表格，RCST 節點可建出如圖 9 所表示的反向通道架構，而時槽的分配則由 TBTP 控制表格敘述，每當 NCC 節點完成反向通道資源分配，NCC 節點將 TBTP 向所有 RCST 節點廣播，各個 RCST 節點可藉由查詢 TBTP 控制表格得知接下來的視框中有哪些時槽是分配給自己。

3.2.2. 用戶端要求訊息

DVB-RCS 標準將反向通道頻寬分配方式分成以下幾種：

- CRA（Continuous Rate Assignment）
- RBDC（Rate Based Dynamic Capacity）
- VBDC／AVBDC（Volume Based Dynamic Capacity／Absolute Volume Based Dynamic Capacity）
- FCA（Free Capacity Assignment）

CRA：

CRA 是一種用戶端與系統服務者簽約書上協定的一種頻寬分配，一旦 RCST 登入 DVB-RCS 網路，CRA 就會持續提供固定大小的頻寬（Bandwidth）給該 RCST，這種資源是自動分配，不需要 RCST 節點發出要求訊息。即使 RCST 節點並沒有資料要傳送，CRA 分配方式依舊會將頻寬分配出去，所以這種分配可能造成反向通道的頻寬浪費。CRA 分配通常用於需要固定而且受保證的頻寬、最小延遲（Delay）、以及最小延遲差異（Delay Jitter）的資料流（Traffic Flow）。

RBDC：

RBDC 是一種動態的頻寬分配方式，RCST 節點週期性量測使用者送出資料的速率並發出要求訊息（此類要求稱爲「RBDC 要求」）到 NCC 節點，送到 NCC 節點的 RBDC 要求會維持一段有效期，直到被新的 RBDC 要求覆蓋或是期限終止。

VBDC／AVBDC：

VBDC 與 RBDC 一樣是一種動態的頻寬分配方式，RCST 節點週期性量測緩衝空間內的資料量，並發出要求訊息（此類要求稱爲「VBDC 要求」）到 NCC 節點，VBDC 要求是累加性的，也就是說，RCST 節點發出的 VBDC 要求會加到之前發出的要求。VBDC 要求不受到保證，若 NCC 節點在做排程時沒有足夠的頻寬可滿足某一個 RCST 節點的 VBDC 要求，NCC 會將不足的數量記錄下來，我們把 NCC 節點爲各個 RCST 節點記錄的不足數量稱爲「VBDC 積欠量」。AVBDC 與 VBDC 相似，唯一差別在於 AVBDC 要求不具有累加性，每一個 AVBDC 要求覆蓋掉之前發出的 VBDC／AVBDC 要求，AVBDC 通常用在 RCST 節點發現所發出的 VBDC 要求遺失時。

FCA：

NCC 節點完成上述幾種分配方式之後如果有剩餘的頻寬，依據系統服務者制訂的政策，NCC 節點可決定是否要將這些頻寬分配出去，將剩餘頻寬分配出去就叫做 FCA。這些頻寬如果沒有分配出去就會造成閒置，而 RCST 節點利用 FCA 分配的頻寬可以減少等待的時間，所以 FCA 分配的目的是提高反向通道使用率（Utilization）以及減少延遲。FCA 與 CRA 一樣是自動分配，不需要 RCST 節點發出要求訊息。

上述幾種頻寬分配中，RBDC、VBDC、以及 AVBDC 都需要 RCST 節點向 NCC 節點發出要求訊息，RCST 節點發出要求訊息直到收到回應的這段期間稱爲「要求延遲」。如 3.2.1 節提到的，每一個 RCST 節點會在分配給自己的要求時槽內，將這些要求裝載在 SYNC Burst 送出。

以下提出在我們設計中，RCST 節點幾個與頻寬分配相關的參數：

- CRA_MAX_RATE：RCST 節點的 CRA 額度，指的是 CRA 分配中可拿到的固定頻寬。
- RBDC_MAX_RATE：RCST 節點的 RBDC 額度，指的是 RBDC 分配中可拿到的最大頻寬。在我們設計中，RBDC 要求是被保證的，也就是說，RCST 節點的 RBDC

要求只要低於 RBDC_MAX_RATE，所發出的要求一定被滿足。

- RBDC_TIMEOUT：此參數代表的是 RBDC 要求的有效期，可以是 1 至 15 個超視框。
- VBDC_MAX_RATE：此參數代表的是 RCST 節點可拿到的頻寬上限。VBDC 要求不受保證，所以即使 RCST 節點發出的 VBDC 要求低於 VBDC_MAX_RATE，所發出的要求也不一定被滿足。

3.3. 協定堆設計

模組化有利於讓研究者在一個系統中開發他們感興趣的主題。在 NCTUns 網路模擬器中，我們以模組方式開發及模擬各式各樣的協定，比如我們有 ARP 模組、AODV 模組等。因此，使用 NCTUns 的研究者可以修改或置換他們感興趣的模組，做爲自己的研究平台。

爲了支援 DVB-RCS 網路模擬，我們在 NCTUns 上開發了一些新的模組。圖 10 顯示六個新開發的節點以及個別對應的模組，圖中深色的是專爲 DVB-RCS 網路系統新增的模組，其餘的是原有的且經過長期嚴格測試的模組。圖 11 以及 12 個別顯示使用者資料在正向通道以及反向通道上所流經的模組。以下子章節中，我們針對新開發節點上各個模組做說明。

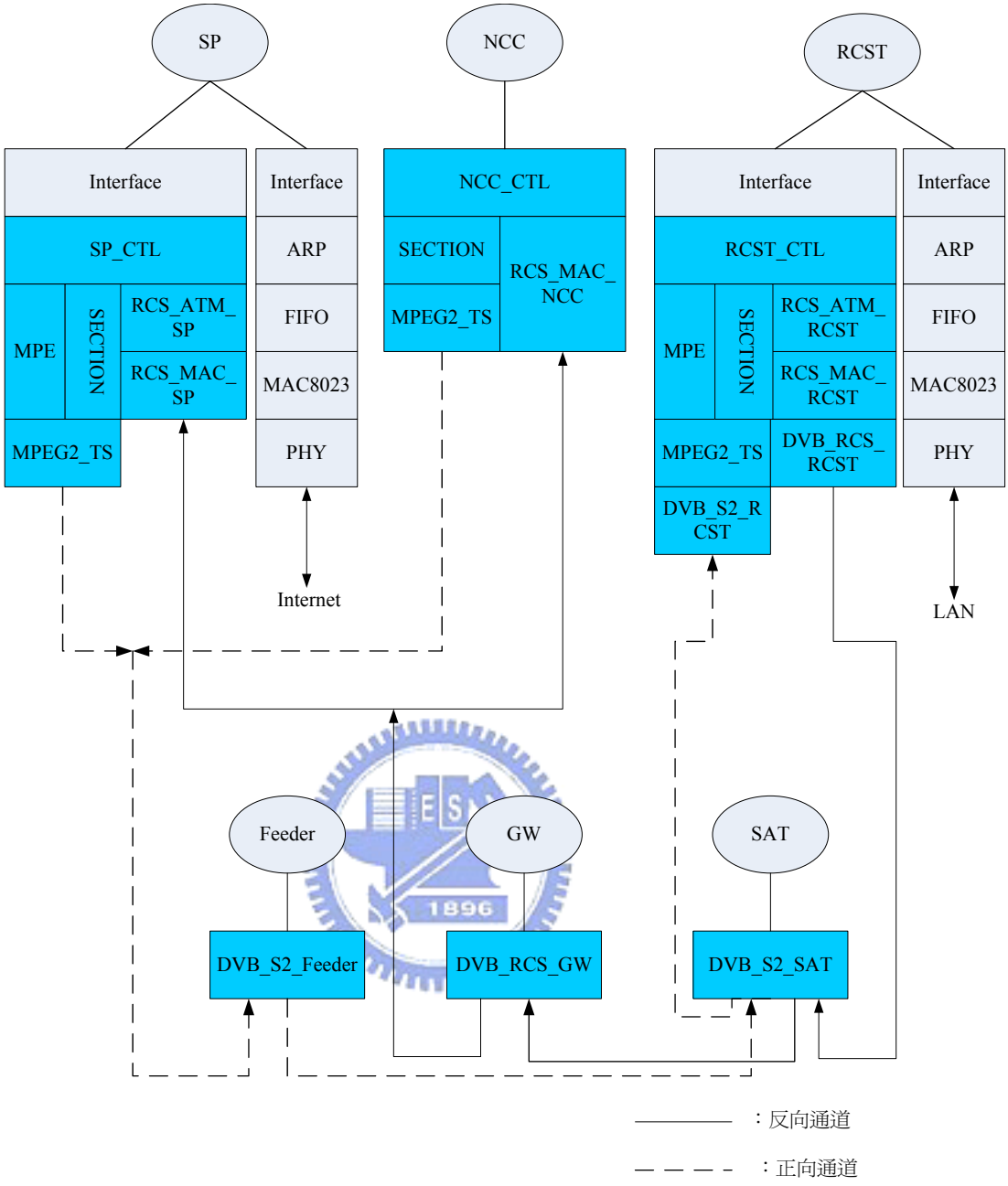


圖 10 六個 DVB-RCS 節點以及對應的協定堆

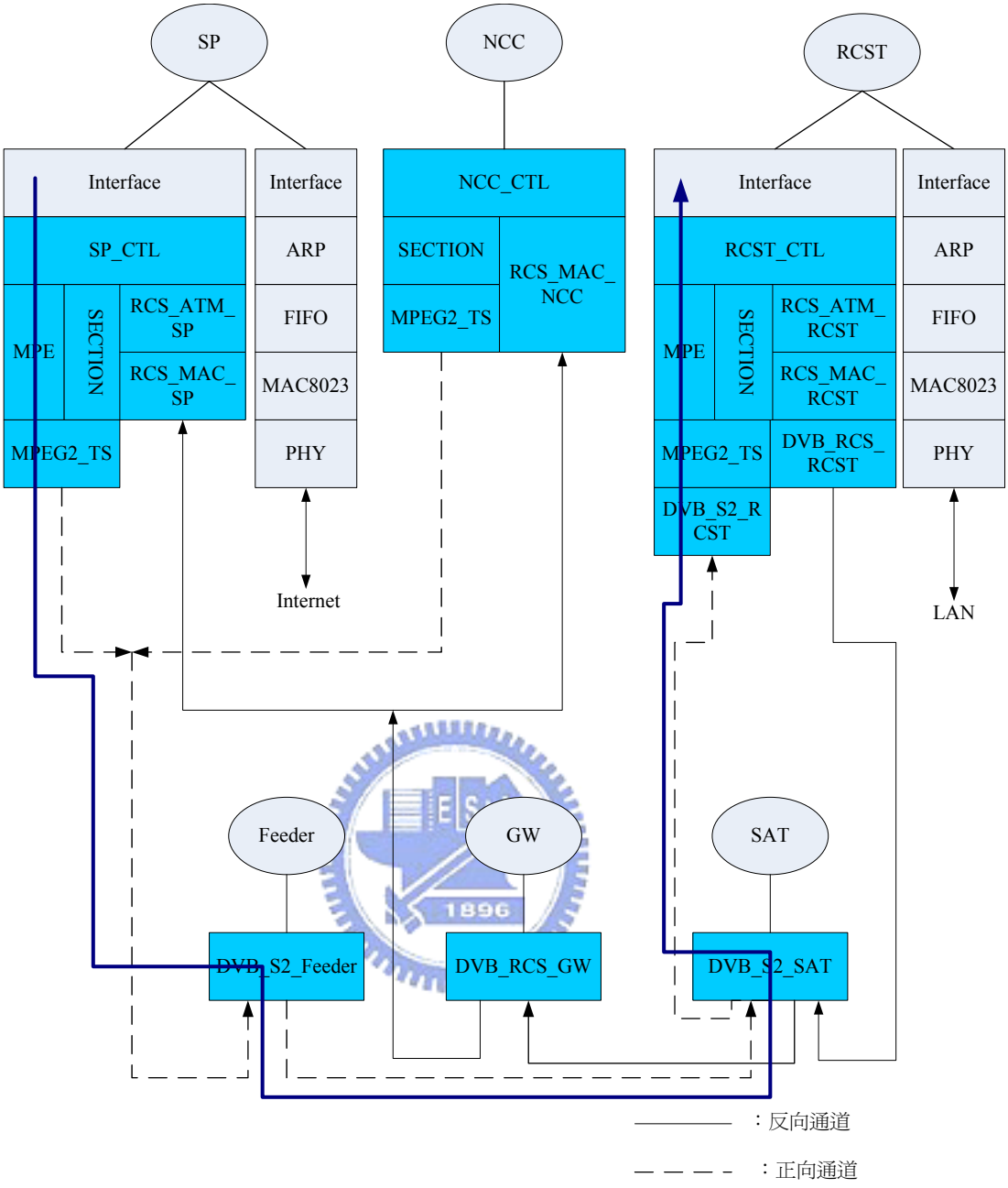


圖 11 使用者資料在正向通道上經過的模組

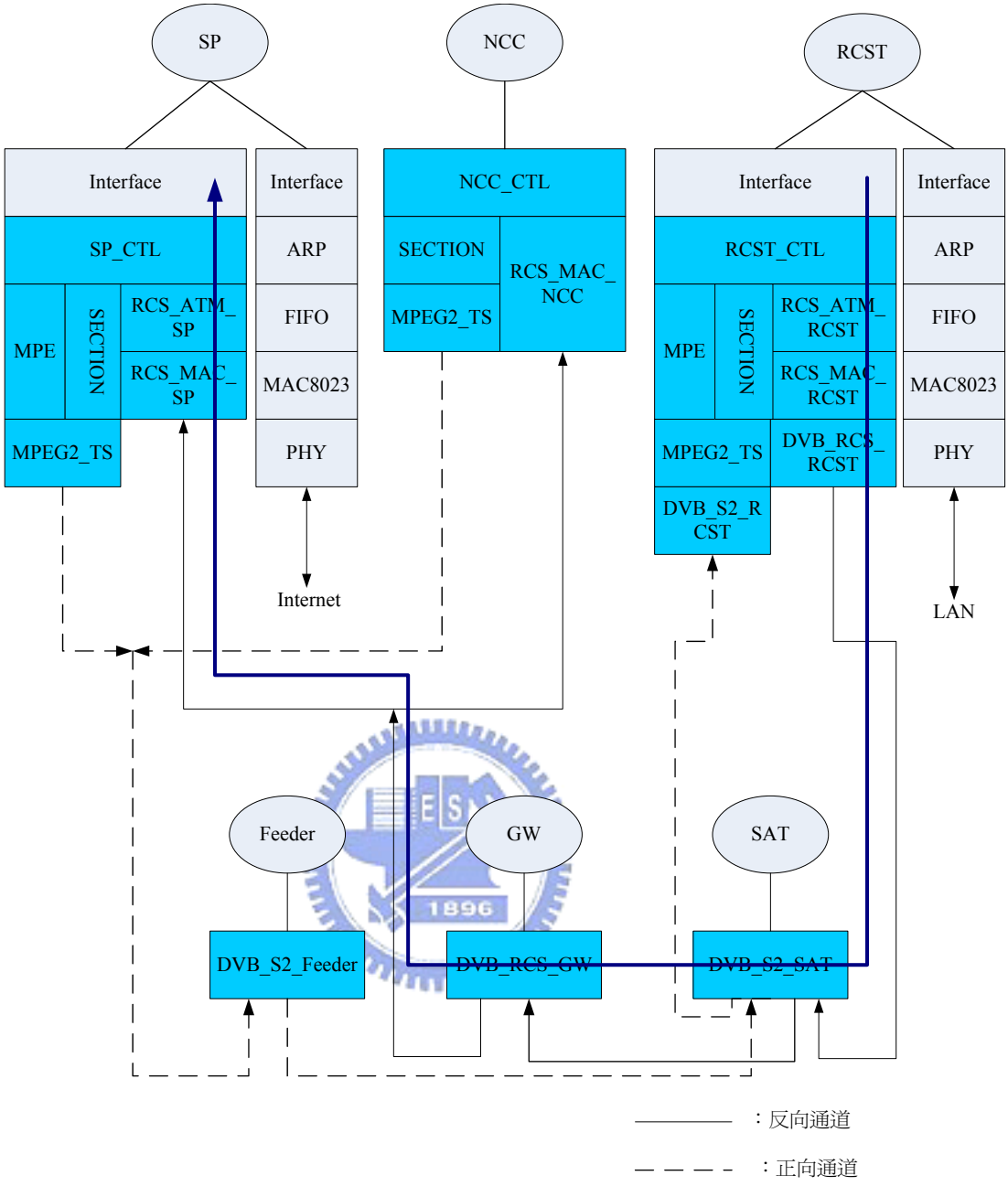


圖 12 使用者資料在反向通道上經過的模組

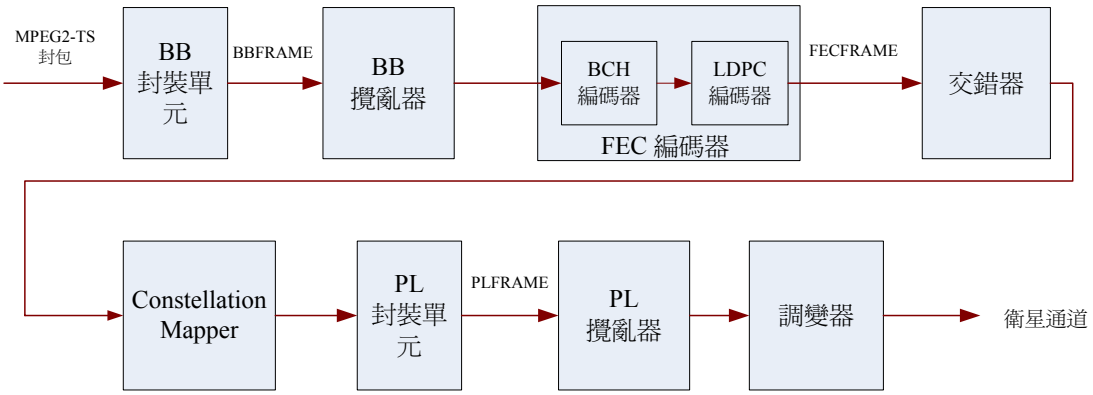


圖 13 正向通道上的通道編碼

3.3.1. SAT 節點上的模組

DVB_S2_SAT 模組

在這一版本的設計上，衛星是一個單純的中繼器（Repeater），所以當 DVB_S2_SAT 模組收到任何下層的資料，對資料不做任何改變，依據資料類型將資料轉送到 GW 節點或網路中所有 RCST 節點。

除了轉送資料，DVB_S2_SAT 模組還負責模擬訊號從衛星到達地面所經過的傳輸延遲（Propagation Delay），傳輸延遲的計算爲衛星與地面距離除上光速。所以 DVB_S2_SAT 模組收到資料時會經過延遲後才做轉送動作。

3.3.2. FD 節點上的模組

DVB_S2_Feeder 模組

圖 13 顯示 DVB-RCS 標準中描述的正向通道實體層 [9]，DVB_S2_Feeder 模組負責模擬正向通道實體層行爲，以下我們依序描述此模組的行爲。

一旦從 MPEG2_TS 模組收到 MPEG2-TS 封包，DVB_S2_Feeder 模組中的 BB（Base Band）封裝單元先將 MPEG2-TS 封包封裝成稱爲 BBFRAME 的格式。首先 BB 封裝單元會對每一個 MPEG2-TS 封包做 CRC-8 編碼，MPEG2-TS 標頭的第一個位元組稱爲同步位元組（Synchronization Byte），這個位元組在 DVB-RCS 正向通道中沒有作用，所以 BB 封裝單元將計算出的 CRC-8 Code 取代這個位元組，最後將 CRC-8 編碼後的封包放入到一個緩衝區暫放。DVB_S2_Feeder 模組每隔一段時間會做傳送動作，此時 BB 封裝單元從上述緩衝區擷取一段資料並加上一個標頭，產生一個 BBFRAME 後傳遞給 BB 攪亂器（Scrambler）。

收到 BBFRAME，BB 攪亂器將 BBFRAME 打散，以避免連串的零或一。經過攪亂的 BBFRAME 會通過 BCH（Bose-Chaudhuri-Hocquenghem）編碼器而後 LDPC（Low

Density Parity Check）編碼器，形成稱爲 FECFRAME 的資料區塊。接下來交錯器（Interleaver）把 FECFRAME 內的資料交錯擺置以減少連串錯誤帶來的影響。由於軟體無法模擬信號層級的行爲，所以從 Constellation Mapper 之後的單元都沒有實作，我們只計算調變出來的符號（Symbol）量並模擬其所需花費的傳輸時間。

最後 DVB_S2_Feeder 模組將產生出的 FECFRAME 轉送至 DVB_S2_SAT 模組，爲了模擬訊號從 FD 節點到達 SAT 節點所需的傳輸延遲，此轉送會被延遲執行。

3.3.3. GW 節點上的模組

DVB_RCS_GW 模組

DVB_RCS_GW 模組把在反向通道收到由衛星送下來的封包做通道解編碼（Channel Decoding）以及反亂碼（De-randomization），經過這些處理後會還原成一個 Burst，之後就會將其送到對應的 NCC 節點以及 SP 節點。在我們的系統設計中，我們把 GW、NCC、SP、以及 FD 四個節點視爲緊密相連的設備，所以忽略彼此之間的傳輸延遲。因此 DVB_RCS_GW 模組一旦解出 Burst 就立刻傳遞給 NCC 節點以及 SP 節點，不模擬傳輸延遲。

3.3.4. NCC 節點上的模組

NCC_CTL 模組

NCC 節點主要的功能是管理整個 DVB-RCS 網路中所有的資源，NCC_CTL 模組便是負責這個任務的模組。NCC_CTL 下掛有兩串不同協定堆，正向通道上的協定堆是用來傳送控制表格，而反向通道上的是用來接收 RCST 節點送來的要求訊息。

在 3.2.1 節中有提到，爲了讓網路中所有 RCST 節點知道如何利用反向通道，NCC 會定期廣播 SCT、FCT、TCT、以及 TBTP 四種控制表格。因此，NCC_CTL 模組負責維護 SCT、FCT、以及 TCT 三種控制表格，並設置一個計時器，每隔一個超視框就廣播這三種控制表格。除此之外，NCC_CTL 模組中含有一個單元稱爲分配者（Allocator），分配者依據所有 RCST 節點的要求，分配反向通道的頻寬，並將分配結果透過 TBTP 告知所有 RCST 節點。分配者是 DVB-RCS 系統的核心，我們在 3.4.1 節中將有詳細說明。

SECTION 模組

由於 NCC_CTL 模組所發送的控制表格的大小不一，可能會超過 DVB 規格中所定義的最大 section 長度，所以 NCC 節點上的 SECTION 模組負責將控制表格包裝成一或多個 section。

MPEG2_TS 模組

在正向通道上，無論是控制表格或是使用者資料都將是以 MPEG2-TS 封包格式傳送。控制表格／使用者資料在 SECTION 模組／MPE 模組被封裝成 section 格式，而依照 [6] 中所定義的 MPEG2-TS 封包格式封裝 section 即是 MPEG2_TS 模組的功能。封裝完成後就會將產生的 MPEG2-TS 封包送到 FD 節點。

RCS_MAC_NCC 模組

由於反向通道是使用 MF-TDMA 的存取機制，RCS_MAC_NCC 模組要能模擬同時接收多組頻道訊號的行爲，因此每當收到從 GW 節點送過來的 Burst 時，這個模組設定此 Burst 所用的頻道的計時器來計數接收時間。爲了模擬碰撞的情況，在接收時間結束之前若是又收到來自相同頻道的 Burst，則該頻道上重疊的 Burst 就會被破壞，必須將其丟棄；反之在接收時間結束之前都沒有收到同頻道的訊號，則在接收時間結束時就會將這個 Burst 轉換回 ATM 封包送給 RCS_ATM 模組。

3.3.5. SP 節點上的模組

SP_CTL 模組

SP 節點主要的功能是讓 DVB-RCS 網路中的 RCST 節點能正確地送收 IP 封包，SP_CTL 模組便是負責這個任務的模組。SP_CTL 下掛有兩串不同協定堆，正向通道上的協定堆是用來傳送控制表格以及 IP 封包到 RCST 節點，而反向通道上的是用來接收 RCST 節點送來的 IP 封包。

正向通道方面，在 3.2.1 節中有提到爲了達到正向通道上的多工／解多工，SP 節點會定期發送 PAT、PMT、NIT、以及 INT 控制表格。所以我們在 SP_CTL 模組中維護這四種控制表格，並設置一個計時器，每隔一個超視框就廣播這些控制表格。由於 RCST 節點是藉由 MPEG2-TS 封包上的 PID 值來判斷是否要接收封包，所以 SP_CTL 模組在將 IP 封包送往下層之前，要先知道目的端（RCST 節點）的 PID 值，並指定送出的 MPEG2-TS 封包上的 PID 值。首先，SP_CTL 模組由設定檔取得每一個 RCST 節點以及 RCST 節點後端子網路的 IP 位址，所以當 SP_CTL 模組由 Interface 模組收到一個 IP 封包時，它能由該封包的目的位址查表決定這個封包應該要送到那一個 RCST 節點，以及這個 IP 封包在封裝成 MPEG2-TS 封包後的 PID 值。接下來，SP_CTL 便把 IP 封包連同 PID 值往 MPE 模組傳遞，MPE 模組完成 MPE 封裝後把 MPE section 連同收到的 PID 值傳遞給 MPEG2_TS 模組，MPEG2_TS 模組在封裝 MPEG2_TS 封包的時候便直接將此 PID 值填入標頭中的 PID 欄位，最後 MPEG2_TS 封包經由正向通道到達 RCST 節點時，RCST 節點就可判斷是否要接收封包。

反向通道方面，當 SP_CTL 模組在反向通道這串協定堆收到一個 IP 封包時，它會直接把封包轉收到 Interface 模組，這樣一來 Linux 核心就會依照預先設定好的路由表來

轉送這個封包，若是這個封包的目的位址爲 Internet 網路中的某一台 Host，則這個封包就會被轉到 SP 節點中對應到 Internet 網路的那串協定堆；若是目的位址爲 DVB-RCS 網路中某 RCST 節點後端的 Host，則這個封包就會被轉到 SP 節點中對應到 DVB-RCS 網路的那串協定堆，之後就會依照上一段所敘述的正向通道機制把這個封包送到特定的 RCST 節點。

MPE 模組

SP 節點上的 MPE 模組負責將 IP 封包封裝成 MPE section，爲了簡化設計，我們只實作 2.1.2 中描述的最佳化封裝模式。

SECTION 模組

略。與 NCC 節點中的 SECTION 模組相同。

MPEG2_TS 模組

略。與 NCC 節點中的 MPEG2_TS 模組相同。

RCS_ATM_SP 模組

RCS_ATM_SP 模組負責將 ATM 封包組合成 IP 封包。每當從 RCS_MAC_SP 模組收到一個 ATM 封包，RCS_ATM_SP 模組先對收到的封包做 CRC 檢查以確認封包沒有錯誤。接下來 RCS_ATM_SP 模組檢查 ATM 標頭中的「最後封包標記」，如果該標記顯示這封包是一串 ATM 封包的最後一個，RCS_ATM_SP 模組將這封包連同之前收到的封包集合成一個 AAL5 封包，並對此 AAL5 封包做 CRC 檢查。CRC 檢查通過之後，RCS_ATM_SP 模組從 AAL5 的負載欄位取出 IP 封包，並將 IP 封包送至 SP_CTL 模組。

RCS_MAC_SP 模組

略。與 NCC 節點中的 RCS_MAC_NCC 模組相同。

3.3.6. RCST 節點上的模組

RCST_CTL 模組

RCST_CTL 模組之下掛有兩串不同的協定堆，一串是用來接收正向通道中的控制表格或是 IP 封包，另一串用來透過反向通道發送 IP 封包。此模組的工作主要是：管理 RCST 節點收到的控制表格、將送往反向通道上的資料流做分類、以及發出要求訊息。

反向通道的傳輸資源是由 NCC 節點排程，並利用 TBTP 控制表格將時槽分配表通

知各個 RCST 節點。收到 TBTP 控制表格後，RCST_CTL 模組從表中取出分配到的時槽的編號，接著拿編號查詢 SCT、FCT、TCT 三張表格，取出時槽參數（比如開始時間、頻率、負載種類等），RCST_CTL 模組會紀錄這些參數作爲往後視框中傳送 Burst 的依據，直到時槽過期才將對應時槽參數清除。

在反向通道上，我們以「來源 IP 位址」、「目的 IP 位址」、「來源埠口（Port）」、「目的埠口」、以及協定種類（UDP 或 TCP）五個值綜合起來作爲「資料流辨識碼」，爲了做資源管理，一個資料流會對應到一個 RCS_MAC_RCST 模組中的 ATM 佇列（Queue）。當 RCST_CTL 模組從 Interface 收到一個 IP 封包時，它會拿 IP 封包上的資料流辨識碼到設定檔取得對應的佇列辨識碼，並將此 IP 封包連同查出的辨識碼一併送往 RCS_ATM_RCST 模組，RCS_ATM_RCST 模組將 IP 封包切割成 ATM 封包之後，將 ATM 封包連同收到的佇列辨識碼一併送往 RCS_MAC_RCST 模組，RCS_MAC_RCST 模組便依據佇列辨識碼來將 ATM 封包導入到指定的 ATM 佇列。

RCST_CTL 模組負責發出頻寬要求。每當分配到一個要求時槽，此模組會向 RCS_MAC_RCST 模組詢問頻寬需求量，並依照標準所訂的格式產生要求訊息，在要求時槽開始時間將要求訊息透過下層模組送出。

MPE 模組

RCST 節點上的 MPE 模組與 SP 節點上的 MPE 模組對稱，它負責將 IP 封包從 MPE section 中取出。

SECTION 模組

RCST 模組中的 SECTION 模組負責將 section 組回控制表格。SECTION 模組有緩衝空間用來暫存那些還無法組成控制表格的 section，每當從 MPEG2_TS 模組收到一個 section，就會檢查是否已經可組合一張控制表格，當一個控制表格成功的被組合回來時，SECTION 模組就會把這個控制表格送往 MPE 模組。

MPEG2_TS 模組

RCST 節點中的 MPEG2_TS 模組負責將 MPEG2-TS 封包組合成 section。在 RCST 節點中的 MPEG2_TS 模組，每一個 PID 都有一個對應的緩衝空間，其用途是暫存那些還無法組成一個 section 的 MPEG2-TS 封包，在 MPEG2_TS 模組收到一個 MPEG2-TS 封包時，由該封包的標頭取出 PID，嘗試將這個封包與那些在緩衝空間中，有相同 PID 的 MPEG2-TS 封包組合成一個 section，當一個 section 成功的組合回來時，MPEG2_TS 模組就會把這個 section 送往 SECTION 模組。

RCS_ATM_RCST 模組

RCS_ATM_RCST 模組負責切割與包裝 IP 封包。一旦從 RCST_CTL 模組收到 IP 封

包，RCS_ATM_RCST 模組先將 IP 封包後面加上一個 AAL5 標尾（Trailer），其中包含 CRC32 的訊息，之後把這段資料填充成 48 位元組的倍數，並以 48 位元組爲一個單位切割這個封包。在切割完成後，依據 ATM 規格中的定義爲每一個 48 位元組長的封包加上 5 位元組的 ATM 標頭。在 ATM 標頭中有一個位元稱作「最後封包標記」是用來表示該 ATM 封包是否爲這一串 ATM 封包的最後一個，ATM 標頭中也包含 CRC8 的訊息。

由於要求訊息並不是以 ATM 封包方式傳遞，所以從 RCST_CTL 模組收到要求訊息時，RCS_ATM_RCST 模組不作任何處理，直接將要求訊息傳遞給下層。

RCS_MAC_RCST 模組

RCS_MAC_RCST 模組的主要工作是把一個或多個 ATM 封包封裝成一個 ATM traffic Burst，並在 RCST_CTL 所指定的時槽開始時間把 TRF Burst 送出。反向通道以 ATM 格式來傳送 IP 封包，每一個送往反向通道的 IP 封包都會在 RCS_ATM_RCST 模組被切割並封裝成 ATM 封包，隨後 ATM 封包被送往 RCS_MAC_RCST 模組，ATM 封包進入到 RCS_MAC_RCST 模組並不是立刻被送出，而是被暫放在佇列中等待可利用的時槽。爲了管理頻寬資源以及提供資料流不同的服務品質，我們設計了四種佇列來放置 ATM 封包，分別是「RT」、「VR-RT」、「VR-JT」、以及「JT」，每一個 RCST 節點可以有一或多個佇列，而 RCST 節點送往反向通道的各條資料流會被導入其對應的佇列。

RT 佇列擁有一個參數「CRA_MAX」，此參數代表一個固定大小且受保證的頻寬（RCST 節點中所有 RT 佇列的 CRA_MAX 值的總和不能超過 RCST 節點的 CRA_MAX_RATE），RT 佇列通常用在需要固定頻寬、最小延遲、以及最小延遲差異的資料流。VR-RT 佇列同時擁有 CRA 以及 RBDC 額度，VR-RT 與 RT 佇列同樣擁有參數「CRA_MAX」，不同的是，對 VR-RT 佇列來說，CRA_MAX 表示一個基本頻寬，VR-RT 佇列可以藉由發出 RBDC 要求以取得額外的頻寬，參數「RBDC_MAX」表示此佇列可得的最大額外頻寬，VR-RT 佇列通常用在變速率的資料流。VR-JT 佇列與 VR-RT 佇列相似，不同的地方在於 VR-JT 佇列沒有 CRA 額度，所以通常資料流在 VR-JT 佇列遭遇的延遲會比在 VR-RT 佇列還高。JT 佇列不擁有任何受保證的頻寬額度，JT 佇列必須透過發出 VBDC 要求來取得頻寬，由於我們的 BoD 機制無法保證滿足 VBDC 要求，所以 JT 佇列通常服務可容忍較高延遲差異的資料流。

在我們的設計中有一個稱爲「要求者（Requestor）」的函式單元負責計算 RCS_MAC_RCST 模組中各個佇列的頻寬需求，並負責將 RCST 節點取得的時槽分配到各個佇列，要求者是 DVB-RCS 系統的核心，我們在 3.4.2 節中將有詳細說明。

DVB_S2_RCST 模組

DVB_S2_RCST 模組與 DVB_S2_Feeder 模組相對應，流程與 DVB_S2_Feeder 模組相反，請參考 3.3.2 節。需注意的是，DVB_S2_RCST 模組不負責模擬傳輸延遲而是負責模擬傳輸時間（Transmission Time），而模擬傳輸時間的方法是延遲 MPEG2-TS 封包

向上層模組傳送的動作。

DVB_RCS_RCST 模組

圖 14 顯示 DVB-RCS 的反向通道實體層。其中的 Burst 封裝單元由 RCS_MAC_RCST 模組負責。與正向通道實體層一樣，我們沒有模擬 Constellation Mapper 以及調變器，只計算調變出來的符號量並模擬所需花費的傳輸時間。DVB_RCS_RCST 模組負責其餘的亂碼器（Randomizer）以及 FEC 編碼器。此模組支援具彈性的 FEC 編碼系統，它的 FEC 編碼系統使用里德所羅門碼（RS code）加上迴旋碼（Convolution Code），編碼率可支援 1/2、2/3、3/4、5/6、7/8。此模組還負責模擬傳輸延遲，編碼後產生的資料區塊會被送至 DVB_S2_SAT 模組，爲了模擬訊號從 RCST 節點到達 SAT 節點所需的傳輸延遲，此轉送會被延遲執行。

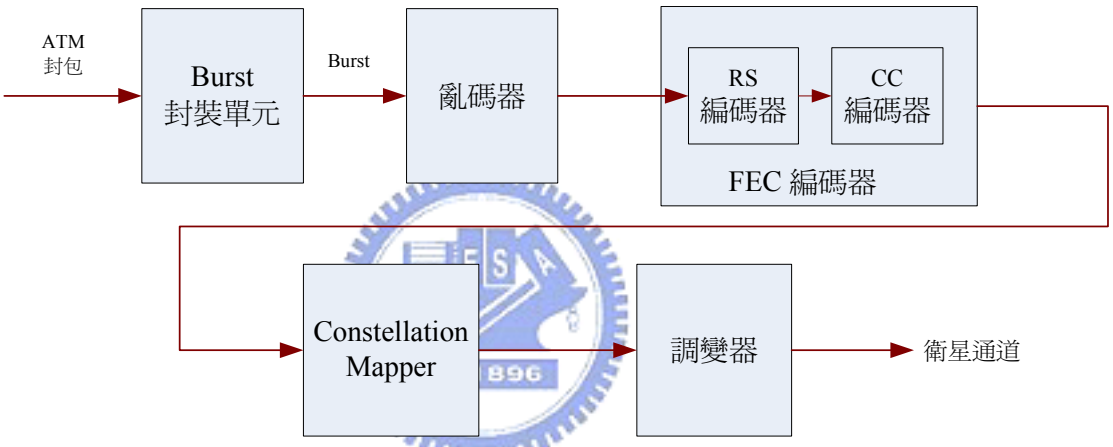


圖 14 反向通道上的通道編碼

3.4. 通道資源分配系統

3.4.1. 分配者

我們在這一節介紹 NCC_CTL 模組中一個稱爲分配者的函式單元，分配者的工作是分配反向通道資源。分配者將所有 RCST 節點送來的要求訊息記錄在一個陣列，我們稱之爲「要求陣列」。在每一個視框開始時間，分配者呼叫內部函式「frame_scheduing( )」來依據要求陣列做資源分配，並產生描述分配結果的 TBTP 控制表格。此函式的實作與使用的分配機制相關，模擬器使用者可以在檔案「timeslot_scheduler.cc」內重新定義自己的分配機制。以下我們用僞碼（Pseudo Code）形式來介紹我們的分配者設計。

在 frame_scheduing( )中，「request timeslot allocation」分配要求時槽；而「CRA allocation」、「RBDC allocation」、「VBDC/AVBDC allocation」、以及「FCA allocation」分配資料時槽。要求時槽的分配使用 Round-Robin 方式，一次分配一個要求時槽給一個

RCST 節點，然後換下一個 RCST 節點，直到所有要求時槽都分配完或者每一個 RCST 節點都有一個要求時槽。在我們的設計中，CRA 以及 RBDC 兩種額度都是受到保證，所以 CRA 以及 RBDC 分配要放置在其他資料時槽分配方式前面；而 FCA 是負責分配剩餘的時槽，所以放在最後面。當時槽分配完畢之後，分配者就會按照分配表產生 TBTP 控制表格，並發送給每一個 RCST 節點。

分配者中使用的資料結構：

```
A[ ] //A[i] stores the total number of TRF timeslot allocated to RCST i.
RV[ ] //RV[i] stores the value of previous RBDC request from RCST i.
VV[ ] //VV[i] stores the backlog of VBDC/AVBDC request from RCST i.
CRA[ ] //CRA[i] stores the number of TRF timeslot to meet the
          //CRA_MAX_RATE of RCST i.
RMAX[ ] //RMAX[i] stores the number of TRF timeslot to meet the
            //RBDC_MAX_RATE of RCST i.
VMAX[ ] //VMAX[i] stores the number of TRF timeslot to meet the
            //VBDC_MAX_RATE of RCST i.
history[ ] //history[i] stores the history of VBDC requests from RCST i.
```

frame_scheduling()

```
 1: for each return link channel Sp do
 2: //assume all of the RCSTs on Sp are RCST 1, RCST 2, ..., RCST n.
 3: reset A[ ] to be a zero vector
 4: do request timeslot allocation for Sp
 5: do CRA allocation for Sp
 6: do RBDC allocation for Sp
 7: do VBDC/AVBDC allocation for Sp
 8: if FCA scheme is turned on then
 9: do FCA allocation for Sp
10: end if
11: end for
```

3.4.1.1. CRA 分配

下面是 CRA 分配的僞碼。CRA 分配中做的事相當簡單，它分配時槽給頻道上擁有 CRA 配額的 RCST 節點。

CRA allocation procedure

```
1: for i = 1 to n do
2: A[i] ← A[i] + CRA[i]
3: end for
```

3.4.1.2. RBDC 分配

下面是 RBDC 分配的僞碼。由於 RBDC 要求具有延續性，所以 RBDC 分配會先將收集到的每一個 RBDC 要求記錄到 RV 陣列，來自相同 RCST 節點的前一次 RBDC 要求會被覆蓋掉。接下來是依據 RV 陣列中尚未過期的 RBDC 要求分配時槽給 RCST 節點。由於各個 RCST 節點在 RBDC 分配中可拿到的頻寬有個上限值，所以在第七行中，分配者對分配出去的時槽量做限制。

RBDC allocation procedure

```
1: for each RBDC request Ri do
2: //assume Ri is submitted from RCST(Ri) and of value value(Ri).
3: RV[RCST(Ri)] ← value(Ri)
4: end for
5: for i = 1 to n do
6: if RV[i] has not expired yet then
7: A[i] ← A[i] + min(RV[i] ,RMAX[i])
8: end if
9: end for
```

3.4.1.3. VBDC 分配

下面是 VBDC 分配的僞碼。由於 VBDC 要求具有累加性，所以 VBDC 分配一開始

先將每一個收到的 VBDC 要求累加到 VBDC 積欠量上；而 AVBDC 要求具有覆蓋性，所以 VBDC 分配會把每一個收到的 AVBDC 要求直接覆蓋掉原先的 VBDC 積欠量。接下來 VBDC 分配以 Round-Robin 方式做時槽分配，輪到的 RCST 節點如果尚有 VBDC 積欠量，而且該 RCST 節點分配到的個數還沒到達上限值的話，就會分配一個時槽給它，接著換下一個 RCST 節點，直到所有的時槽都分配出去或者每一個 RCST 節點的 VBDC 積欠量都清空。

VBDC/AVBDC allocation procedure

```
 1: declare AV[ ] //AV[i] records the number of TRF timeslot allocated to
                  //RCST i in the VBDC/AVBDC allocation.
 2: for each VBDC request Vi do
 3: //assume Vi was submitted from RCST(Vi) and of value value(Vi).
 4: VV[RCST(Vi)] ←VV[RCST(Vi)] + value(Vi)
 5: end for
 6: for all AVBDC request Vi do
 7: //assume Vi was submitted from RCST(Vi) and of value value(Vi).
 8: VV[RCST(Vi)] ← value(Vi)
 9: end for
10: while there exists spare TRF timeslots and VV[ ] is not a zero vector do
11: for i = 1 to n do
12: if there exists no spare TRF timeslots then
13: stop VBDC/AVBDC allocation
14: end if
15: if VV[i] ≠ 0 and AV[i] ≤ VMAX[i] then
16: AV[i] ← AV[i] + 1
17: A[i] ← A[i] + 1
18: VV[i] ← VV[i] − 1
19: end if
20: end for
21: end while
```

3.4.1.4. FCA 分配

FCA 分配的探討是我們這篇論文的主題，爲了觀察 FCA 分配對網路效能的影響，我們設計實作兩種 FCA 機制：FCA-RR 以及 FCA-PR-m。FCA-RR 機制是把剩餘時槽做平均分配；而 FCA-PR-m 是把剩餘時槽依照每一個 RCST 節點發出的 VBDC 要求量做比例性分配。

FCA-RR 分配機制

下面是 FCA-RR 分配的僞碼。FCA-RR 分配以 Round-Robin 方式做時槽分配，一次分配一個時槽給一個 RCST 節點，然後換下一個 RCST 節點。在我們的設計中，FCA 分配的時槽是給 RCST 節點中的 JT 佇列使用，而 JT 是以 VBDC 方式來要求頻寬，所以當 FCA 分配時槽給某個 RCST 節點時，應該減低該 RCST 節點的 VBDC 積欠量。

FCA allocation procedure – RR

```
1: while there exists spare TRF timeslot do
2: for i = 1 to n do
3: if there exists no spare TRF timeslot then
4: stop FCA allocation
5: end if
6: A[i] ← A[i] + 1
7: if VV[i] ≠ 0 then
8: VV[i] ← VV[i] − 1
9: end if
10: end for
11: end while
```

FCA-PR-m 分配機制

FCA-PR-m 依據以往各個 RCST 節點發出的 VBDC 要求量來分配時槽，爲此，分配者使用一個歷史函式（History Function）來紀錄各個 RCST 節點的 VBDC 要求量，下面這段僞碼被安插在 VBDC/AVBDC allocation 之中來實現 VBDC 歷史紀錄，其中 m 值是 0 至 1 之間的權數。

```
1: for each RCST i do
2: history[i] = history[i] * m + VBDC_REQ_VALUE(i) * (1-m)
3: endfor
```

下面是 FCA-PR-m 分配的僞碼。首先分配者先將所有 RCST 節點的 VBDC 歷史值做加總，接著依照 RCST 節點的 VBDC 歷史值佔總值的比例做時槽分配，依照比例分配後如果還有剩餘的時槽，以 Round-Robin 方式分配剩餘時槽。與 FCA-RR 分配相同，當 FCA 分配時槽給某個 RCST 節點時，應該減低該 RCST 節點的 VBDC 積欠量。

FCA allocation procedure – PR-m

```
 1: //assume there are u spare TRF timeslots.
 2: sum_history ← sum of history[ ]
 3: for i = 1 to n do
 4: alloc ← ceil (u * history[i] / sum_history)
 5: A[i] ← A[i] + alloc
 6: if VV[i] ≥ alloc then
 7: VV[i] ← VV[i] − alloc
 8: else
 9: VV[i] ← 0
10: end if
11: if there exists spare TRF timeslots then
12: dispatch spare timeslots in Round-Robin fashion
13: end if
14: end for
```

3.4.2. 要求者

我們在這一節介紹 RCST_CTL 模組中一個稱爲要求者（Requestor）的函式單元，要求者的工作是向 RCS_MAC_RCST 模組內每一個佇列的代理者（Agent）詢問頻寬需求，並將所有需求整理後產生要求訊息。每一個視框計算一次頻寬需求，而每一個佇列的需求量會被個別紀錄，作爲將來分配時槽的依據。要求者還有一項工作是負責將 RCST 節點得到的時槽分配到各個佇列，時槽分配必須參考先前記錄的佇列需求量來執行。在每一個視框開始時間，要求者呼叫內部函式「grant_demand_and_compute_demand( )」來執行上述工作。以下我們用僞碼形式來介紹我們的要求者設計。

在以下程式碼中，我們以 next_demand[i]表示佇列 Qi 的需求量，每當有 ATM 封包進入佇列 Qi 時，next_demand[i]的值會增加。我們以 f 表示將開始的視框的編號、並以 d 表示分配延遲。grant_demand_and_compute_demand( )中，要求者先爲 RT 以及 VR-RT 佇列分配 CRA 配額，接著爲 VR-RT 以及 VR-JT 分配 RBDC 配額，最後才是將剩餘的時槽分配給 JT 佇列。

grant_demand_and_compute_demand( )

```
 1: for each RT or VR-RT queue Qi do
 2: compute_CRA_demand(Qi)
 3: end for
 4: for each RT or VR-RT queue Qi do
 5: grant_CRA_demand(Qi)
 6: end for
 7: for each VR-RT or VR-JT queue Qi do
 8: grant_RBDC_demand(Qi)
 9: end for
10: grant_VBDC_demand( )
11: for each VR-RT or VR-JT queue Qi do
12: compute_RBDC_demand(Qi)
13: end for
14: for each JT queue Qi do
15: compute_VBDC_demand(Qi)
16: end for
17: sum up the capacity requests and generate capacity request message
```

下列僞碼中，CRA_demand[i][f]表示佇列 Qi 期待在視框 f 中取得的時槽個數。在計算時槽需求量時，如果當時佇列的長度不超過佇列的 CRA 額度（CRA_MAX），則佇列會期待在即將開始的視框 f 中取得足夠的時槽來清空整個佇列。相反的，如果當時佇列的長度超過 CRA 額度，則佇列期待拿到符合 CRA 額度的時槽量。

compute_CRA_demand(Qi)

```
1: CRA_demand[i][f] ← min (CRA_MAX[i], next_demand[i])
2: next_demand[i] ← next_demand[i] − CRA_demand[i][f]
3: return zero
```

grant_CRA_demand(Qi)負責分配時槽，如果得到的時槽個數無法滿足 Qi 的期待個數（這種狀況發生在通道錯誤率高使得 TBTP 控制表格遺失時），爲了不讓延遲增加，Qi 會將 ATM 封包丟棄。

grant_CRA_demand(Qi)

```
1: if available_timeslot ≥ CRA_demand[i][f] then
2: dispatch CRA_demand[i][f] timeslots to Qi
3: else
4: drop data of size CRA_demand[i][f] from head of Qi
5: end if
```

grant_RBDC_demand(Qi)負責分配 RBDC 配額，如果時槽數量不足（這種狀況發生在通道錯誤率高使得 TBTP 控制表格或要求訊息遺失時），VR-JT 的處理方式是爲不足的部分重新發出要求，而 VR-RT 佇列的處理方式是丟棄 ATM 封包，以避免延遲增加。

grant_RBDC_demand(Qi)

```
1: if available_timeslot ≥ RBDC_demand[i][f] then
2: dispatch RBDC_demand[i][f] timeslots to Qi
3: else
4: dispatch available_timeslot timeslots to Qi
5: if Qi is a VR-RT queue then
6: drop data of size RBDC_demand[i][f] – available_timeslot from head of Qi
7: else
8: next_demand[i] ← next_demand[i] + RBDC_demand[i][f] – vailable_timeslot
9: end if
10: end if
```

在 compute_RBDC_demand(Qi)中，RBDC_demand[i][f]表示佇列 Qi 期待在視框 f 中取得的時槽個數，由於送出的要求訊息要經過 d 視框後才得到回應，所以分配者在視框 f 中的計算是反應在將來的視框 f+d。如果視框 f 內有機會發送要求訊息，則 Qi 會期待在視框 f+d 中得到符合所發出的訊息的時槽數。如果無法發出要求訊息而且上一次發出的 RBDC 要求還沒過期，則 Qi 會期待得到符合上次要求的時槽數。若是無法送出要求而且上一次發出的 RBDC 要求已經過期，則 Qi 在視框 f+d 將拿不到時槽。

compute_RBDC_demand(Qi)

1: if the RCST has opportunity to send capacity request then
2: RBDC_demand[i][f + d] ← min (VBDC_MAX[i], next_demand[i])
3: else if previous RBDC request has not expired yet then
4: RBDC_demand[i][f + d] ← the value of previous request
5: else
6: RBDC_demand[i][f + d] ← zero
7: end if
8: next_demand[i] ← next_demand[i] − RBDC_demand[i][f + d]
9: return RBDC_demand[i][f+d]

JT 佇列中，我們把 ATM 封包分成以下三類：

1. 已經爲其發出要求而且發出時間已經超過分配延遲
2. 已經爲其發出要求但發出時間尚未超過分配延遲
3. 尚未爲其發出 VBDC 要求的封包

分配者爲 JT 佇列分配時槽時，是依照各個佇列中，三種封包的數量。因爲第一類的封包是三種封包中最早進入 RCST 節點的，所以分配者爲 JT 佇列分配時槽時，應該先分給第一類的封包使用，也就是說，分配者會先統計每一個佇列中的第一類的封包量，將時槽先分配給那些擁有第一類的封包的佇列。爲第一類封包分配時槽後如果還有剩餘時槽（表示 RCST 節點因 FCA 機制取得多餘的時槽，或者是因爲 RCST 節點中具有 CRA 配額的佇列並沒有將全部配額用盡），接著分給第三類封包，最後是第二類封包。我們以 next_demand[Qi]表示佇列 Qi 中第一類封包的總和、以 sum_expired_demand[Qi]表示第二類的總和、並以 sum_non_expired_demand[Qi]表示第三類的總和。

grant_VBDC_demand( )負責分配時槽給 RCST 節點中的各個 JT 佇列，首先是取 sum_expired_demand[ ]作爲分配上限，以 Round-Robin 方式分配，也就是說一次分配一個時槽給一個佇列，每當輪到佇列 Qi 時，如果 Qi 已經拿到 sum_expired_demand[Qi]個時槽，則不分配給 Qi，直接換下一個佇列。接著取 next_demand[ ]作爲分配上限，同樣以 Round-Robin 方式分配，最後取 sum_non_expired_demand[ ]作爲分配上限，以 Round-Robin 方式分配。

grant_VBDC_demand( )

```
1: compute sum_expired_demand[ ] and sum_non_expired_demand[ ]
2: dispatch timeslots in Round-Robin fashion depending on
   sum_expired_demand[ ]
3: if available_timeslot ≠ 0
4: dispatch timeslots in Round-Robin fashion depending on next_demand[ ]
5: endif
6: if available_timeslot ≠ 0
7: dispatch timeslots in Round-Robin fashion depending on
   sum_non_expired_demand[ ]
8: endif
```

要求者對 VBDC 的處理與對 RBDC 一樣有做要求訊息遺失的處理，在我們的設計中，判斷 VBDC 要求遺失是依靠 TBTP 控制表格中，一個稱爲「queue_empty」的欄位，若 RCST 節點 R1 在 NCC 節點的 VBDC 積欠量爲零，TBTP 控制表格上就會將 R1 的 queue_empty 設爲 true，當 R1 收到 TBTP 控制表格，發現此欄位爲 true，而且 R1 中有 JT 佇列的 sum_expired_demand 不爲零，則表示有要求訊息遺失。上述狀況發生時，要求者會爲歸類爲第一類的 ATM 封包重新要求時槽。

compute_VBDC_demand(Qi)

```
1: if request loss happened then
2: next_demand[i] ← next_demand[i] + sum_expired_demand[i]
3: end if
4: if the RCST has opportunity to send capacity request then
5: VBDC_demand[i][f + d] ← next_demand[i]
6: else
7: VBDC_demand[i][f + d] ← zero
8: end if
9: next_demand[i] ← zero
10: return VBDC_demand[i][f+d]
```

四、 DVB-RCS 模擬及正確性驗證

4.1. 系統測試相關參數

我們在這一章中以模擬方式驗證系統的正確性，我們把模擬所使用的重要系統參數列在表 2 以及表 3，並將使用的拓樸顯示在圖 15。表 3 僅列出共通的參數，額外的參數將於個別模擬範例中描述。

參數名稱	參數值	參數描述
P_{UDP}	1472 位元組	UDP 封包的負載大小
R_S	16 Mboud	符號傳輸率
S_{FEC}	Normal（8100 位元組）	經過 FEC 編碼後的區塊大小
調變模式	QPSK	
LDPC 編碼率	1/3	

表 2 正向通道相關參數

參數名稱	參數值	參數描述
P_{UDP}	1472 Bytes	UDP 封包的負載大小
R_S	9.935 Mboud	符號傳輸率
R_{CC}	1/2	CC 編碼率
N_{ATM}	2	一個 TRF Burst 裝載的 ATM 封包個數
N_{TB}	98	一個視框中的資料時槽個數
N_{RB}	2	一個視框中的要求時槽個數
L_P	16 Symbols	Preamble Length
T_G	$2 * 10^{-7}$ sec	Guard Time
T_T	10^{-4} sec	時槽長度
F_T	10^{-2} sec	視框長度
分配延遲	50 視框	RCST 節點發出要求訊息與收到回應之間的時間
頻道總頻寬	7.5 Mbits/sec	反向通道上每一個頻道的傳輸率上限值
調變模式	QPSK	

表 3 反向通道相關參數

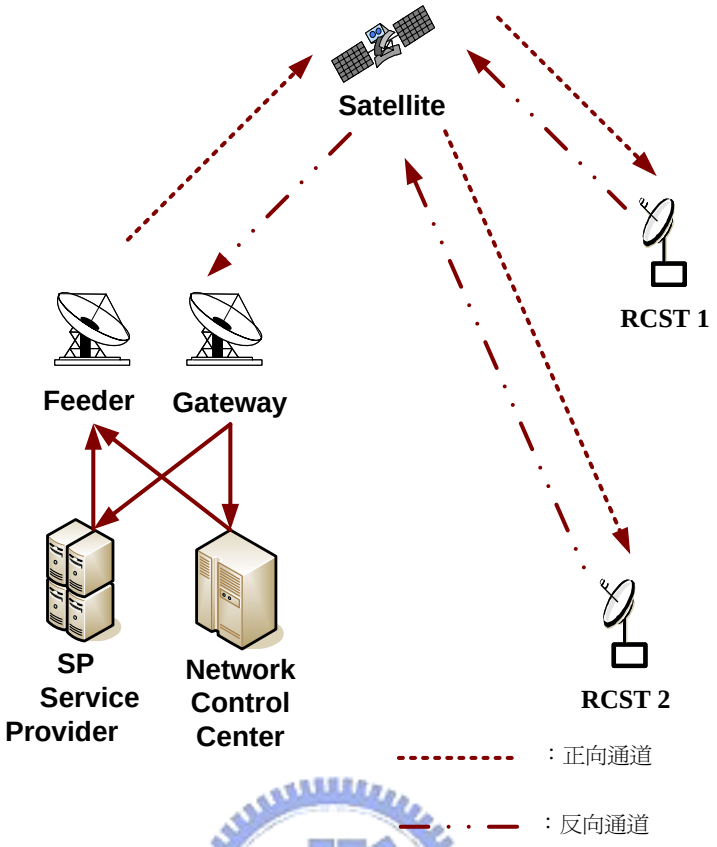


圖 15 模擬使用的拓樸

4.2. UDP 模擬數據與驗證

4.2.1. 正向通道

在這一節中，我們推算應用層在正向通道上可得傳輸率的理論值，並且在我們開發的 DVB-RCS 模擬平台上做模擬，將量測出來的數據與理論值比較。

理論值推導

在正向通道上有三個主要的額外負擔來源：控制表格、通道編碼、以及封裝。以下爲了簡化計算，我們忽略控制表格所帶來的額外負擔。爲了方便數據推導，我們將封包的封裝過程整理至圖 16，詳細說明請參考第三章。

我們以 A_{PL} 表示每秒可傳送 PLFRAME 的平均個數、以 A_{BB} 表示每秒可傳送 BBFRAME 的平均個數、以 A_{TS} 表示每秒可傳送 MPEG2-TS 封包的平均個數、以 A_{MPE} 表示每秒可傳送 MPE section 的平均個數、以 T_{APP} 表示應用層可拿到的傳輸率。以下我們依序推算 A_{PL}、A_{BB}、A_{TS}、A_{MPE}、以及我們所要的「T_{APP}」。

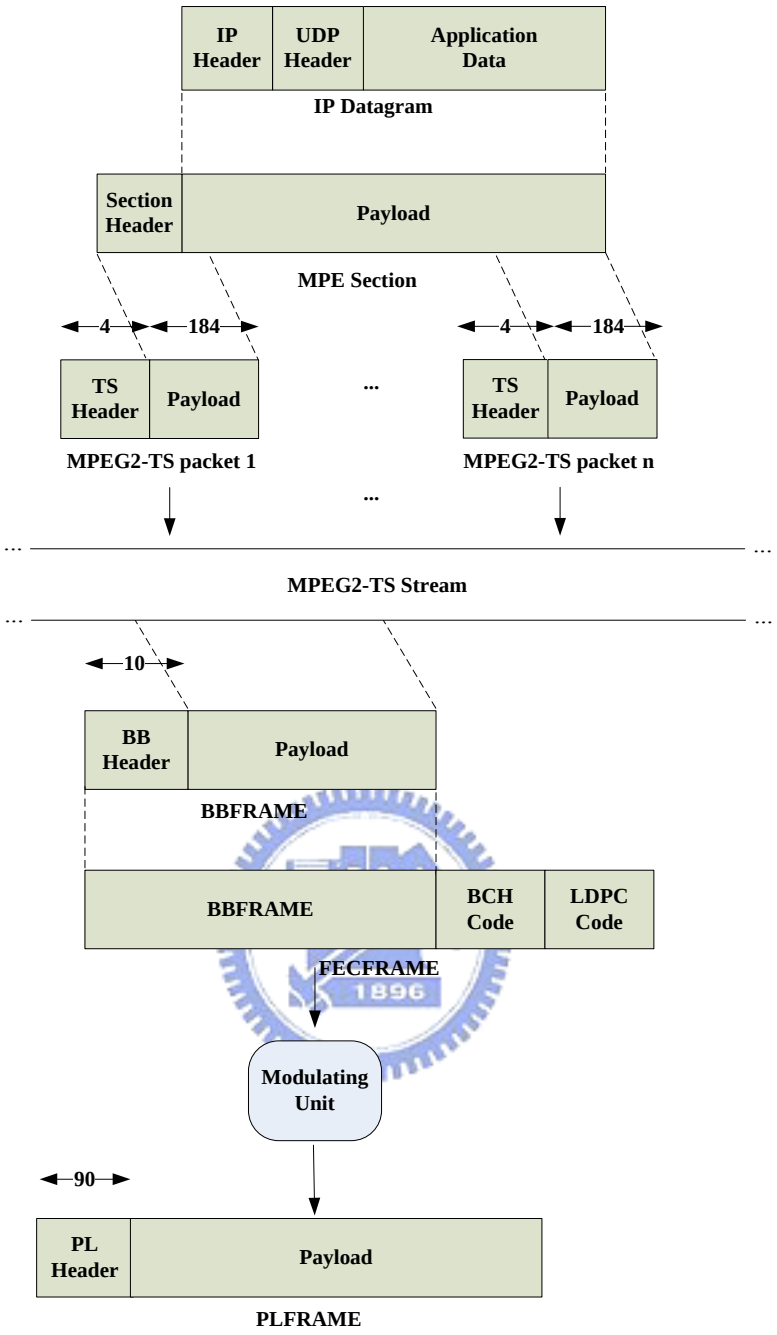


圖 16 正向通道的資料封裝過程

我們選用的 QPSK 調變模式可用一個符號表示 2 位元資料，所以一個 FECFRAME 經過調變會成爲 $S_{FEC} * 8 / 2$ 個符號並當作 PLFRAME 的負載。一個 PLFRAME 附帶一個大小爲 90 個符號的標頭，所以 PLFRAME 大小 S_{PL} 爲：

$$S_{PL} = S_{FEC} * 8 / 2 + 90$$
$$= 8100 * 8 / 2 + 90$$
$$= 32490 \text{ Symbols}$$

實體層每秒可以傳輸 R_S 個符號，所以我們可以推得：

$$A_{PL} = R_S / S_{PL}$$
$$= 16 * 10^6 / 32490$$
$$= 492.459 / sec$$

一個 PLFRAME 攜帶一個 FECFRAME，而且一個 FECFRAME 攜帶一個 BBFRAME，所以 $A_{BB}=A_{PL}$。爲了算 A_{TS}，我們先算出一個 BBFRAME 平均可以裝載的 MPEG2-TS 封包個數。表 4 爲 DVB-RCS 標準書中通道編碼的參數，我們選用 1/3 的 LDPC 編碼率以及 8100 位元組的 FECFRAME，從表中我們可知道未編碼區塊大小（也就是 BBFRAME 大小）爲 2676 位元組，而 BBFRAME 標頭大小爲 10 位元組，所以可以算出 BBFRAME 負載大小 S_{BB} 爲 2686 位元組。MPEG2-TS 封包大小固定爲 188 位元組，而 BBFRAME 在封裝 MPEG2-TS 封包時可以將 MPEG2-TS 封包切割成兩份放置在不同 BBFRAME 送出，以減少填塞帶來的額外負擔。所以一個 BBFRAME 平均可以裝載 $S_{BB} / 188$ 個 MPEG2-TS 封包，我們可以推得：

$$A_{TS} = A_{BB} * S_{BB} / 188$$
$$= 492.459 * 2686 / 188$$
$$= 7.036 * 10^3 / sec$$

Normal FECFRAME				Short FECFRAME			
LDPC rate	**Uncoded block**	**LDPC coded block**	**BCH coded block**	**LDPC rate**	**Uncoded block**	**LDPC coded block**	**BCH coded block**
1/4	2001	2025	8100	1/4	384	405	2025
1/3	2676	2700	8100	1/3	654	675	2025
2/5	3216	3240	8100	2/5	789	810	2025
1/2	4026	4050	8100	1/2	879	900	2025
3/5	4836	4860	8100	3/5	1184	1215	2025
2/3	5380	5400	8100	2/3	1329	1350	2025
3/4	6051	6075	8100	3/4	1464	1485	2025
4/5	6456	6480	8100	4/5	1554	1575	2025
5/6	6730	6750	8100	5/6	1644	1665	2025
8/9	7184	7200	8100	8/9	1779	1800	2025
9/10	7274	7290	8100	9/10	N/A	N/A	2025

表 4 FEC 編碼前以及編碼後的區塊大小

（單位：位元組）

接下來我們推算一個 MPE section 需要幾個 MPEG2-TS 封包來傳送。我們以 H_{UDP} 表示 UDP 標頭大小、以 H_{IP} 表示 IP 標頭大小、以 H_{MPE} 表示 MPE section 標頭大小。一個 MPE section 裝載一個 IP 封包，UDP 標頭大小爲 8 位元組、H_{MPE} 在我們的設計中爲 16 位元組，最常見的 IP 標頭大小爲 20 位元組，因此 MPE section 的大小 S_{MPE} 爲：

$$\begin{aligned} S_{MPE} &= H_{IP} + H_{UDP} + P_{UDP} + H_{MPE} \\ &= 20 + 8 + 1472 + 16 \\ &= 1516 \text{ Bytes} \end{aligned}$$

一個 MPEG2-TS 封包可容納 184 位元組的負載，所以一個 MPE section 經過切割後，所形成 MPEG2-TS 封包的數量 N_{ST} 可表示爲：

$$\begin{aligned} N_{ST} &= \text{ceil}(S_{MPE} / 184) \\ &= \text{ceil}(1516 / 184) \\ &= 9 \end{aligned}$$

每一秒可傳送 A_{TS} 個 MPEG2-TS 封包，而每 N_{ST} 個 MPEG2-TS 封包可傳送一個 MPE section，所以每一秒可傳送的 MPE section 個數爲：

$$\begin{aligned} A_{MPE} &= A_{TS} / N_{ST} \\ &= 7.036 * 10^3 / 9 \\ &= 7.818 * 10^2 \text{ / sec} \end{aligned}$$

一個 MPE section 裝載一個 IP 封包，而一個 IP 封包裝載 1472 位元組來自應用層的資料，所以應用層傳輸率爲：

$$\begin{aligned} T_{APP} &= A_{MPE} * 1472 \\ &= 7.818 * 10^2 * 1472 \\ &= 1.151 * 10^6 \text{ Bytes/sec} \\ &= 9.208 \text{ Mbits/sec} \end{aligned}$$

模擬測試

爲了測得應用層的傳輸率，我們設定一組簡單的網路拓樸，如同圖 15 所示。我們在 SP 與 RCST 1 之間建立一條 UDP 連線，SP 設定一個 Greedy UDP 送端程式，Greedy UDP 送端程式會盡其可能地利用頻寬，我們並在 RCST 1 設定 UDP 收端程式。我們對各種的實體層模式做模擬測試，所量測出來的應用層傳輸率顯示在表 5，對應的理論值也列在表中，理論值與量測值的差異來自於控制表格的額外負擔以及計算時四捨五入產生的誤差。

Normal FECFRAME			Small FECFRAME		
PHY Mode	**理論傳輸率**	**模擬傳輸率**	**PHY Mode**	**理論傳輸率**	**模擬傳輸率**
QPSK-1/4	6.824	6.562	QPSK-1/4	5.085	4.822
QPSK-1/3	9.208	9.009	QPSK-1/3	8.756	8.493
QPSK-2/5	10.988	10.726	QPSK-2/5	10.592	10.328
QPSK-1/2	13.765	13.503	QPSK-1/2	11.816	11.551
QPSK-3/5	16.541	16.278	QPSK-3/5	16.099	15.833
QPSK-2/3	18.405	18.142	QPSK-2/3	17.934	17.669
QPSK-3/4	20.705	20.443	QPSK-3/4	19.770	19.503
QPSK-4/5	22.093	21.830	QPSK-4/5	20.993	20.727
QPSK-5/6	23.032	22.770	QPSK-5/6	22.217	21.950
QPSK-8/9	24.588	24.325	QPSK-8/9	24.053	23.785
QPSK-9/10	24.897	24.633	QPSK-9/10	N/A	N/A
8PSK-3/5	24.777	24.514	8PSK-3/5	24.016	23.749
8PSK-2/3	27.570	27.306	8PSK-2/3	26.754	26.487
8PSK-3/4	31.015	30.751	8PSK-3/4	29.492	29.224
8PSK-5/6	34.501	34.237	8PSK-5/6	33.143	32.875
8PSK-8/9	36.832	36.567	8PSK-8/9	35.882	35.612
8PSK-9/10	37.294	37.030	8PSK-9/10	N/A	N/A
16APSK-3/4	41.296	41.030	16APSK-3/4	39.109	38.626
16APSK-4/5	44.064	43.799	16APSK-4/5	41.530	41.261
16APSK-5/6	45.937	45.672	16APSK-5/6	43.951	43.681
16APSK-8/9	49.041	48.776	16APSK-8/9	47.582	47.311
16APSK-9/10	49.660	49.391	16APSK-9/10	N/A	N/A
32APSK-3/4	51.549	51.282	32APSK-3/4	48.623	48.343
32APSK-4/5	55.004	54.737	32APSK-4/5	51.632	51.352
32APSK-5/6	57.342	57.075	32APSK-5/6	54.642	54.360
32APSK-8/9	61.217	60.949	32APSK-8/9	59.156	58.873
32APSK-9/10	61.984	61.717	32APSK-9/10	N/A	N/A

表 5 正向通道上，不同實體層模式對應的傳輸率

（單位：Mbits/sec）

4.2.2. 反向通道

在這一節中，我們推算應用層在反向通道上可得傳輸率的理論值，並且在我們開發的 DVB-RCS 模擬平台上做模擬，將量測出來的數據與理論值比較。

理論值推導

爲了方便數據推導，我們將封包的封裝過程整理至圖 17。在以下的推算中，首先我們會推算一個 IP 封包被切割出的 ATM 封包個數（以下我們以 N 表示這個數值），接著推算時槽長度 T_T。一個 TRF Burst 可裝載 N_{ATM} 個 ATM 封包，所以一個 IP 封包平均需要 N / N_{ATM} 個 TRF Burst 來傳送。把 IP 封包上的應用層資料量除上 N / N_{ATM} 個時槽長度，最後扣除傳送要求訊息的額外負擔即可得到應用層傳輸率。

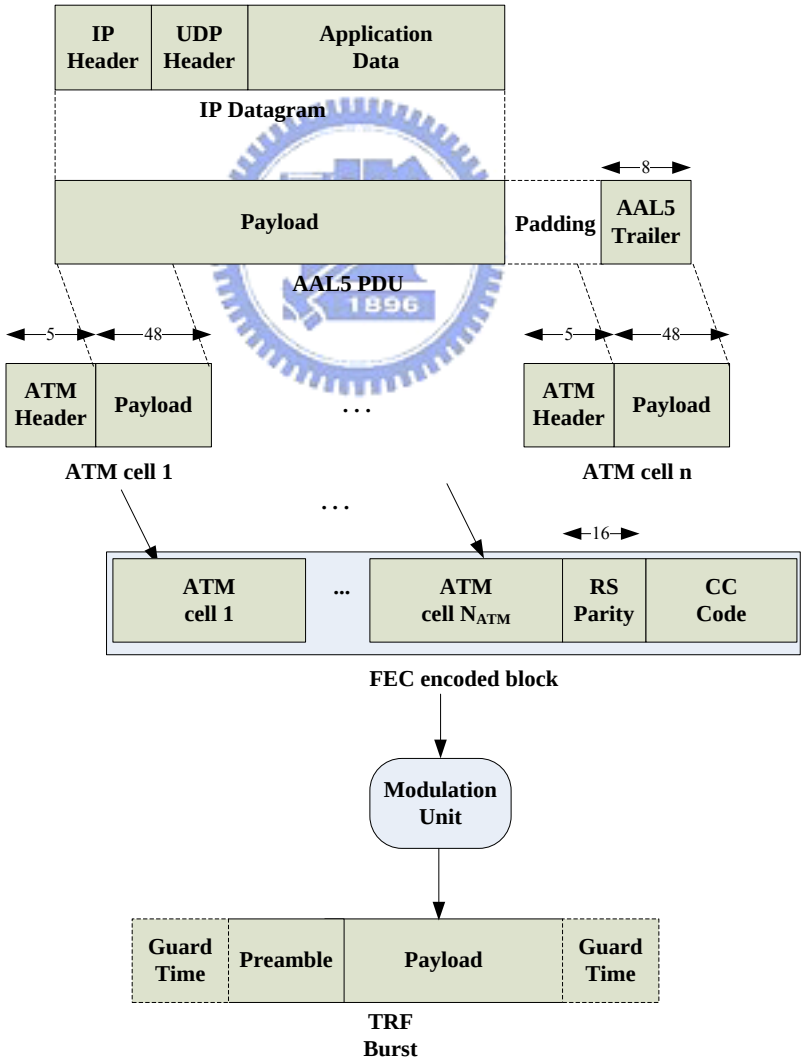


圖 17 反向通道的資料封裝過程

以下我們以 H_{UDP} 表示 UDP 標頭大小、以 H_{IP} 表示 IP 標頭大小、以 H_{AAL5} 表示 AAL5 標尾大小、以 S_{AAL5} 表示 AAL5 封包大小、以 P_{ATM} 表示 ATM 封包中負載的大小、並以 T_{APP} 表示我們要算的應用層傳輸率。在 AAL5／ATM 協定的封裝過程中，IP 封包會被加上 AAL5 標尾、填充成 48 位元組的倍數、並以 48 位元組爲一個單位切割成 ATM 封包的負載。UDP 標頭大小爲 8 位元組，AAL5 標頭大小爲 8 位元組，最常見的 IP 標頭大小爲 20 位元組。因此，一個 IP 封包切割出的 ATM 封包個數可表示爲：

$$\begin{aligned} N &= \text{ceil}\left((P_{UDP} + H_{IP} + H_{UDP} + H_{AAL5}) / P_{ATM}\right) \\ &= \text{ceil}\left((1472 + 20 + 8 + 8) / 48\right) \\ &= 32 \end{aligned}$$

我們以「FEC 區塊」表示經過 FEC 編碼後的資料區塊，而 S_{FEC} 代表 FEC 區塊大，並以 S_B 表示 TRF Burst 的大小，接下來我們會先推算這兩個值。一個 FEC 區塊裝有 N_{ATM} 個 ATM 封包，每一個 ATM 封包的大小爲 53 位元組，所以 FEC 區塊上的負載量爲 $53 * N_{ATM}$ 位元組。RS Parity 大小固定爲 16 位元組，再加上 CC 編碼帶來的額外負擔，FEC 區塊大小可表示爲：

$$\begin{aligned} S_{FEC} &= (53 * N_{ATM} + 16) / R_{CC} \\ &= (53 * 2 + 16) / (1/2) \\ &= 244 \text{ Bytes} \end{aligned}$$

反向通道使用 QPSK 調變模式，而 QPSK 調變模式下的一個符號可攜帶兩個位元，所以 FEC 區塊經過調變之後會成爲 SFEC * 8 / 2 個符號。調變後的訊號串前面加上同步訊號（preamble）即成爲 TRF Burst，所以 TRF Burst 長度可表示爲：

$$\begin{aligned} S_B &= S_{FEC} * 8 / 2 + L_P \\ &= 244 * 8 / 2 + 16 \\ &= 992 \text{ Symbols} \end{aligned}$$

我們知道符號傳送率表示每秒可傳送的符號個數，所以一個 TRF Burst 所花費的傳送時間爲 S_B / R_S 秒。爲了降低 Burst 之間的訊號干擾，Burst 之間需要加上一段保衛時間（Guard Time），所以一個時槽長度可表示成：

$$\begin{aligned} T_T &= S_B / R_S + T_G \\ &= 992 / (9.935 * 10^6) + 2 * 10^{-7} \\ &= 1.000 * 10^{-4} \text{ Sec} \end{aligned}$$

一個 IP 封包平均需要 N / N_{ATM} 個 TRF Burst 來傳送，而一個 TRF Burst 需花費 T_T 秒，所以傳送一個 IP 封包需要 $T_T * N / N_{ATM}$ 秒。一個 IP 封包內裝有 P_{UDP} 位元組應用層資料，所以可得傳輸量爲 $P_{UDP} / (T_T * N / N_{ATM})$。一個視框由 N_{TB} 個資料時槽以及 N_{RB} 個要求時槽組成，所以視框中傳輸資料的比例爲 $N_{TB} / (N_{TB} + N_{RB})$，因此實際傳輸量爲：

$$\begin{aligned} T_{APP} &= P_{UDP} / (T_T * N / N_{ATM}) * N_{TB} / (N_{TB} + N_{RB}) \\ &= 1472 / (1.000 * 10^{-4} * 32 / 2) * 98 / (98 + 2) \\ &= 9.016 * 10^5 \text{ Bytes/sec} \\ &= 7.213 \text{ Mbits/sec} \end{aligned}$$

以上所算出的應用層傳輸率比表 3 中的頻道總頻寬還小，這是因爲我們以單位時間內收到的 ATM 負載量作爲頻道總頻寬；而以單位時間內收到的 UDP 負載量作爲應用層傳輸率。兩個值之間的差異來自於 AAL5 標尾以及 UDP/IP 標頭帶來的額外負擔。

模擬測試

爲了測得應用層的傳輸率，我們設定一組簡單的網路拓樸，我們在 SP 與 RCST 1 之間建立一條 UDP 連線，RCST 1 設定一個 Greedy UDP 送端程式，並在 SP 設定 UDP 收端程式。如表 6 所列，我們在 RCST 1 上設置一個 JT 佇列，爲了讓 RCST 1 取得最大的頻寬，我們將 VBDC_MAX_RATE 設爲頻道總頻寬，由於只有一條連線，所以此佇列將擁有整個頻道的頻寬。我們對各種的 CC 編碼率做模擬測試，所量測出來的應用層傳輸率顯示在表 7，對應的理論值也列在表中，理論值與量測值的差異來自於四捨五入引起的誤差。

佇列種類	JT
佇列長度	11000 個 ATM 封包
頻寬分配參數	CRA_MAX_RATE = 0 Mbits/sec RBDC_MAX_RATE = 0 Mbits/sec RBDC_TIMEOUT = 1 VBDC_MAX_RATE = 7.5Mbits/sec
FCA 機制	無

表 6 4.2.2 節所用到的模擬參數

CC 編碼率	**理論傳輸率**	**測得傳輸率**
1/2	7.209	7.213
2/3	9.555	9.568
3/4	10.717	10.722
5/6	11.872	11.876
7/8	12.447	12.459

表 7 反向通道上，不同 CC 編碼率對應的傳輸率

（單位：Mbits/sec）

佇列種類	JT
佇列長度	11000 個 ATM 封包
頻寬分配參數	CRA_MAX_RATE = 0 Mbits/sec RBDC_MAX_RATE = 0 Mbits/sec RBDC_TIMEOUT = 1 VBDC_MAX_RATE = 7.5Mbits/sec
FCA 機制	FCA-RR／FCA-PR-0.8

表 8 4.3 節所用到的模擬參數

UDP 1 設定檔	**UDP 2 設定檔**
1: type: udp	**1:** type: udp
2: start_time: 1.2005	**2:** start_time: 1.2005
3: on-off: 1	**3:** on-off: 1
4: on: time: 0.01 const 0.0015 length: const 924	**4:** on: time: 0.01 const 0.0017 length: const 924
5: on: time: 0.01 const 0.0015 length: const 924	**5:** on: time: 0.01 const 0.0015 length: const 924
6: on: time: 0.01 const 0.0015 length: const 924	**6:** on: time: 0.01 const 0.0011 length: const 924
7: on: time: 0.01 const 0.0015 length: const 924	**7:** on: time: 0.01 const 0.0008 length: const 924
8: off: time: 1000	**8:** off: time: 1000
9: end	**9:** end

表 9 4.3 節所用到的 UDP 設定檔

4.3. FCA 行爲驗證

在我們的模擬平台中，我們實作了兩種 FCA 機制，分別是 FCA-RR 以及 FCA-PR-m，在這節中，我們藉由兩組模擬範例來檢驗我們的實作。如表 8 所示，除了使用不同的 FCA 機制外，這兩組模擬範例使用相同的參數設定。爲了展示 FCA 機制，在每一個範例中，設有兩條 UDP 連線，UDP 1 由 RCST 1 送往 SP；而 UDP 2 由 RCST 2 送往 SP，每條 UDP 都由一個對應的 JT 佇列服務。表 9 顯示兩條 UDP 連線的設定檔，以下我們以 UDP 1 說明我們的設定檔。設定檔第一行指定使用的協定種類爲 UDP，第二行指定連線從 1.2005 秒開始，第三行指定以下的設定重複次數，數值 1 表示不重複。第四行表示在 0.01 秒內，每隔 0.0015 秒產生一個 924 位元組的 UDP 封包。我們特意將每一行的間隔設爲與視框長度相同，使得第四行對應到第 120 視框，第五行對應到第 121 視框，後面行數以此類推。由於視框長度（0.01 秒）是封包產生週期（0.0015 秒）的六點多倍，所以在第 120 視框內會有 7 個 UDP 封包產生。一個 924 位元組的 UDP 封包被切割成 20

個 ATM 封包，而每一個時槽可以放置 2 個 ATM 封包，所以 7 個 UDP 封包總共產生 140 個 ATM 封包並且需要 70 個時槽來傳送。在 NCC 收到第一個要求訊息之前，無論是 FCA-RR 或 FCA-PR-m 機制都以平均方式分配視框中的 98 個資料時槽，所以兩個 RCST 節點各取得 49 個資料時槽，在第 120 視框中 RCST 1 需要 70 個時槽，所以 RCST 1 在取得並利用 49 個時槽後，爲缺少的 21 個時槽發出 VBDC 要求。從第 120 視框到第 123 視框兩個 RCST 節點的時槽需求量以及發出的 VBDC 要求量顯示在表 10。而在表 11 中，我們列出經過分配延遲後對應的時槽分配，下面我們對表 11 上的數據作解釋。

FCA-RR 機制

我們以第 120 視框爲例，在此視框中 RCST 1 要求 21 個時槽而 RCST 2 要求 11 個時槽，所以 NCC 上的分配者在 VBDC 分配階段會個別分配 21 以及 11 個時槽，FCA 分配階段將剩下來的 66 個時槽平均分配。因此，在分配延遲之後的第 170 視框中，RCST 1 收到 21+ 66 / 2 = 54 個時槽，而 RCST 2 收到 11 + 66 / 2 = 44 個時槽。第 171 視框以及第 172 視框的分配結果也可以相同方式算出。然而，在第 123 視框中，VBDC 要求的總和（102）大於一個視框中的資料時槽總數，VBDC 分配階段分配 21 個時槽給 RCST 1 以及 77 個時槽給 RCST 2，VBDC 分配後沒有剩餘的時槽，所以 21 以及 77 即爲兩個 RCST 節點在第 173 視框中所得的時槽個數。

視框編號	時槽需求量		VBDC 要求量	
	RCST 1	RCST 2	RCST 1	RCST 2
120	70	60	21	11
121	70	70	21	21
122	70	90	21	41
123	70	130	21	81

表 10 RCST 節點的時槽需求個數

視框編號	FCA-RR		FCA-PR-0.8	
	RCST 1	RCST 2	RCST 1	RCST 2
170	54	44	65	33
171	49	49	52	46
172	39	59	37	61
173	21	77	21	77

表 11 RCST 節點分配到的時槽個數

FCA-PR-0.8 機制

以第 120 視框爲例，在此視框中 RCST 1 要求 21 個時槽而 RCST 2 要求 11 個時槽，VBDC 分配階段個別分配 21 以及 11 個時槽後，FCA 分配階段將剩下來的 66 個時槽依據 VBDC 要求量做分配。RCST 1 要求 21 個時槽、RCST 2 要求 11 個時槽，所以 history[RCST 1]變成 0 * 0.8 + 21 * 0.2 = 4.2、history[RCST 2]變成 0 * 0.8 + 11 * 0.2 = 2.2。FCA 分配過程如下：

1. 分配 ceil(66 * 4.2 / (4.2+2.2)) = 43 時槽給 RCST 1
2. 分配 ceil(66 * 2.2 / (4.2+2.2)) = 22 時槽給 RCST 2
3. 將最後一個時槽以 Round-Robin 分配並給了 RCST 1

因此 RCST 1 共取得 21+43+1=65 個時槽，而 RCST 2 共取得 11+22=33 個時槽。第 171 視框以及第 172 視框的分配結果也可以相同程序算出。在第 123 視框的分配與 FCA-RR 模擬範例的第 123 視框相同，不再贅述。

4.4. TCP 公平性

TCP 是 IP 網路中最重要的協定之一，許多受歡迎的應用，比如 WWW 以及 E-mail 等，都是建立在 TCP 之上。爲了不增加網路阻塞且達到好的連線效能，TCP 使用緩起步（Slow Start）、流量控制（Flow Control）、以及壅塞控制（Congestion Control）等技巧。在這一節中，我們藉由模擬來顯示 TCP 在 DVB-RCS 網路中呈現公平性，模擬範例的拓樸如圖 15。以下我們分別對正向通道以及反向通道測試 TCP 公平性。

4.4.1. 正向通道

爲了測試 TCP 在正向通道上的公平性，我們在 SP 設置兩個 Greedy TCP 送端程式，Greedy TCP 送端程式會盡其可能地利用頻寬，RCST 節點上各設置一個 TCP 收端程式，TCP i 表示 SP 送到 RCST i 的 TCP 連線，連線時間從第 1 秒到 100 秒。圖 18 顯示模擬結果，從圖中可看出，以長期來看，TCP 連線在 DVB-RCS 網路的正向通道上具有公平性。

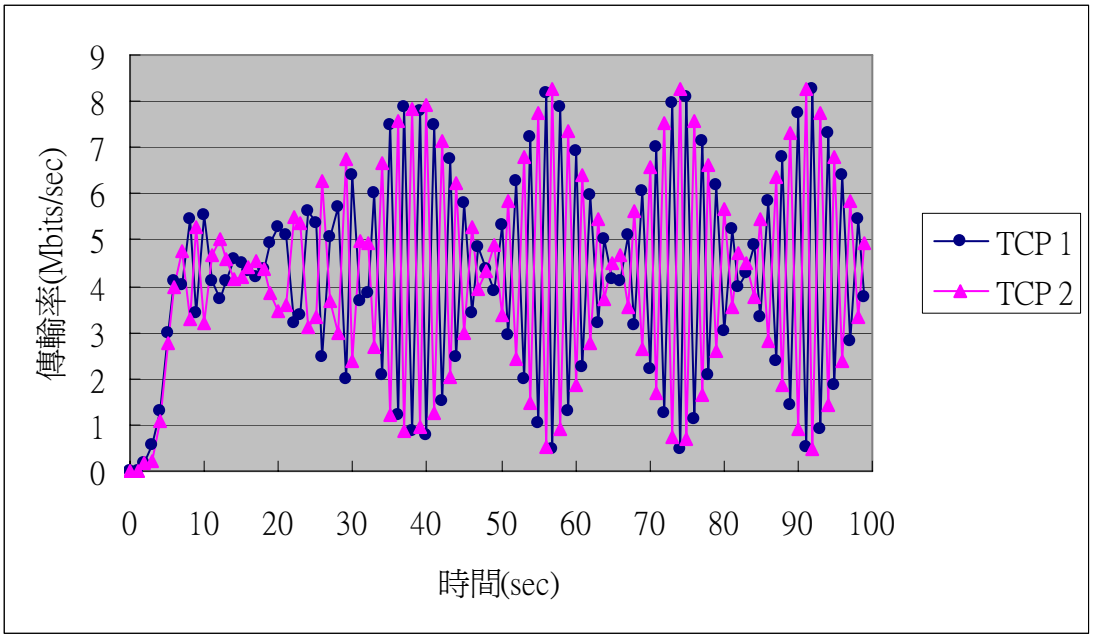


圖 18 TCP 在正向通道上的公平性

佇列種類	JT
佇列長度	11000 個 ATM 封包
頻寬分配參數	CRA_MAX_RATE = 0 Mbits/sec RBDC_MAX_RATE = 0 Mbits/sec RBDC_TIMEOUT = 1 VBDC_MAX_RATE = 2.3 Mbits/sec
FCA 機制	無

表 12 4.4.2 節所用到的模擬參數

4.4.2. 反向通道

我們在這一節測試無 FCA 機制之下，反向通道上的 TCP 公平性。我們在 SP 設置兩個 TCP 收端程式，RCST 節點上各設置一個 Greedy TCP 送端程式，TCP i 表示 RCST i 送到 SP 的 TCP 連線，連線時間從第 1 秒到 100 秒。由於 CRA 以及 RBDC 配額是被保證，使用 RT、VR-RT、VR-JT 佇列的資料流可得到的反向通道資源並不會受其他資料流的影響，因此我們只對 JT 佇列做探討，表 12 顯示此模擬範例用到的模擬參數。圖 19 顯示模擬結果，從圖中可看出，以長期來看，TCP 連線在 DVB-RCS 網路的反向通道上具有公平性。

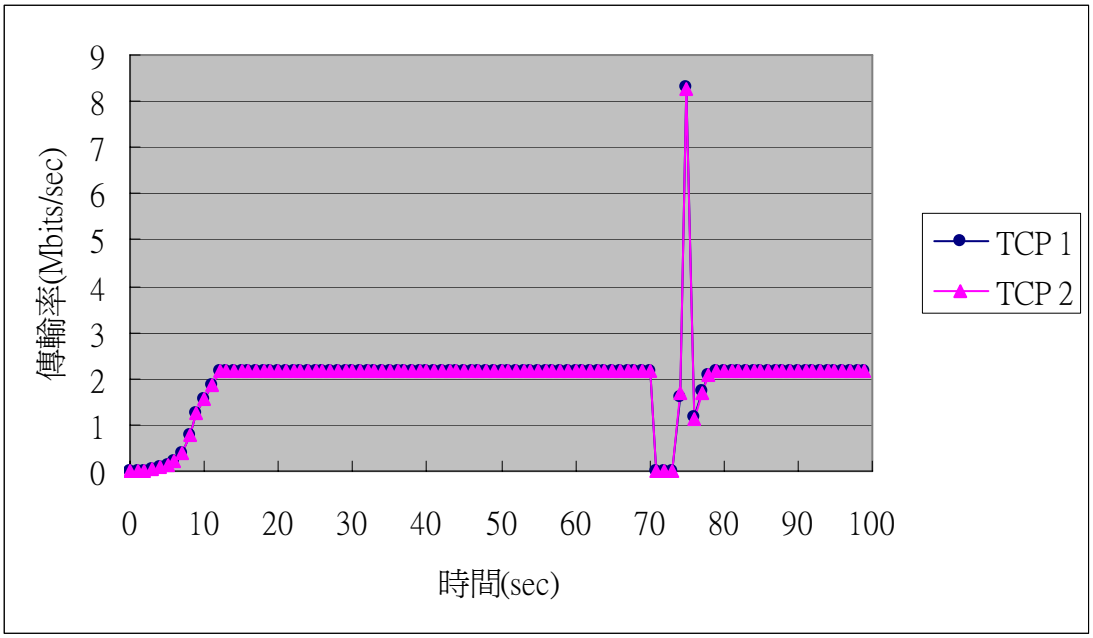


圖 19 TCP 在反向通道上的公平性

五、 FCA 對 TCP 效能的影響

5.1. TCP 公平性

佇列種類	JT
佇列長度	11000 個 ATM 封包
頻寬分配參數	CRA_MAX_RATE = 0 Mbits/sec RBDC_MAX_RATE = 0 Mbits/sec RBDC_TIMEOUT = 1 VBDC_MAX_RATE = 2.3 Mbits/sec
FCA 機制	FCA-RR／FCA-PR-0.8

表 13 5.1 節所用到的模擬參數

在 4.4.2 節中提到，在無 FCA 機制之下， TCP 連線在反向通道上具有公平性。而在這一節中，我們觀察當 FCA 機制開啓時，反向通道是否仍舊保有 TCP 公平性。我們設定兩個模擬範例來測試兩種 FCA 機制是否影響 TCP 公平性，兩個範例中各有兩個 RCST 節點，RCST 節點上各設置一個 Greedy TCP 送端程式，在 SP 設置兩個 TCP 收端程式，TCP i 表示 SP 送到 RCST i 的 TCP 連線，連線時間從第 1 秒到 100 秒。使用的模擬參數列在表 13，除了 FCA 機制個別改成 FCA-RR 以及 FCA-PR-0.8 外，其餘參數與 4.4.2 節用到的相同。圖 20 顯示使用 FCA-RR 機制之下，兩條 TCP 連線的傳輸率；而圖 21 顯示使用 FCA-PR-0.8 機制之下的傳輸率。從圖中可看出，兩條 TCP 連線平分反向通道頻寬，兩種 FCA 機制都不影響 TCP 的公平性。

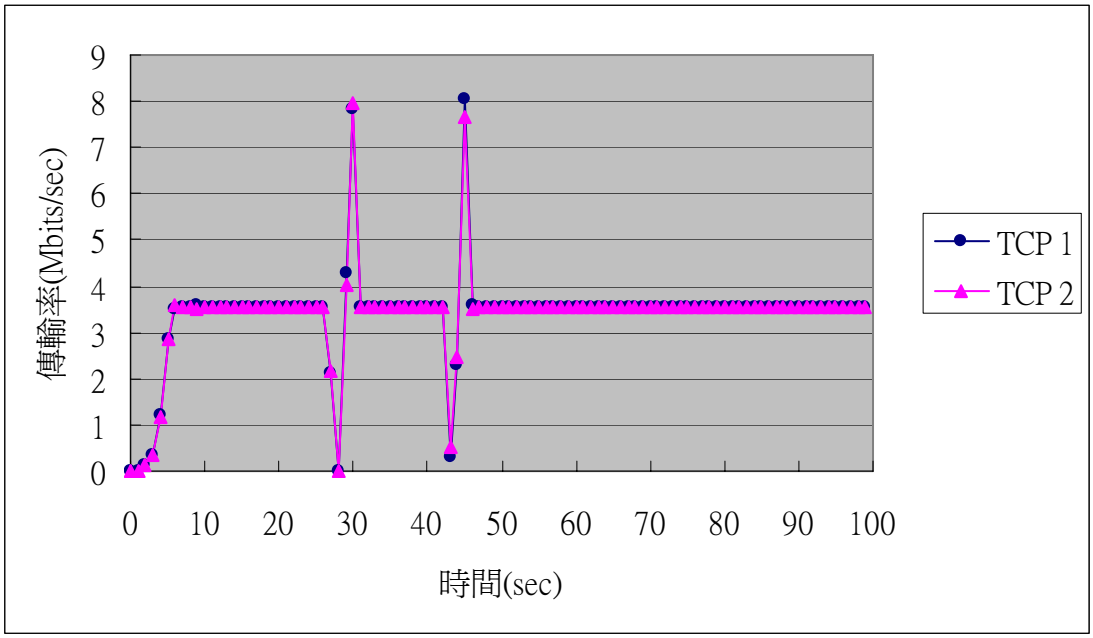


圖 20 FCA-RR 機制下的 TCP 公平性

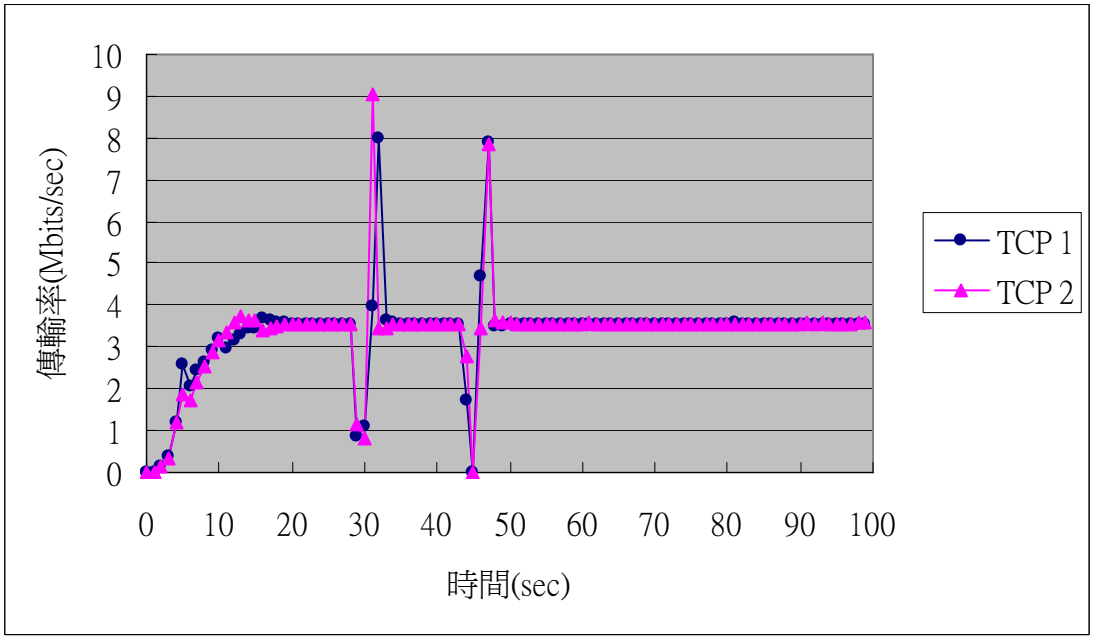


圖 21 FCA-PR-0.8 機制下的 TCP 公平性

在這兩張數據圖中都可以觀察到一個現象，在 30 秒以及 45 秒左右傳輸率突然下降，緊接著突然上升，接下來回到平衡狀態，這個現象起因於 TCP 壅塞控制。TCP 會嘗試增大壅塞視窗（Congestion Window），當 TCP 發送速度大於下層的可傳輸率時，TCP 封包就會在下層被丟棄。TCP 送端依靠收端的 ACK 回應判斷是否有封包遺失，一旦送端察覺封包遺失，送端就會啓動重送機制，在重送封包到達收端之前所收到的 Out-of-Order 封包會被堆放在 TCP 收端緩衝空間，直到重送封包到達並補滿坑洞時，這些封包才一併送給應用層。圖中的傳輸率下降就是因爲發生封包遺失，而傳輸率上升是因爲坑洞被補滿，此時送端的壅塞視窗大小已經下降，TCP 便回復穩定狀態。

5.2. TCP 起步

在上一節中顯示，無論使用 FCA-RR 或 FCA-PR-0.8 機制，兩條同時啓動的 TCP 連線公平地平分頻寬。而在這一節中，我們觀察與探討兩種 FCA 機制下，TCP 連線的起步狀況。爲了觀察 TCP 連線起步，我們同樣設立兩條 RCST 節點往 SP 的 TCP 連線，TCP 1 連線在第 1 秒啓動，到了第 15 秒時，TCP 1 已經到達最大傳輸率並保持傳輸率穩定，此時啓動才 TCP 2 連線。除了連線啓動時間不同外，其餘參數與上一節相同，請參考表 13。圖 22 顯示使用 FCA-RR 機制之下，兩條 TCP 連線的傳輸率；而圖 23 顯示使用 FCA-PR-0.8 機制之下的傳輸率。

首先我們觀察 10 至 15 秒的數據，在 FCA-PR-0.8 機制下，TCP 1 的最大傳輸量相當接近頻道總頻寬（7.5Mbits/sec）；而 FCA-RR 機制下，雖然頻道上只有 TCP 1 一條連線，TCP 1 還是無法取得最大頻道頻寬，VBDC_MAX_RATE 設爲 2.3 Mbits/sec，TCP 1 在 VBDC 分配時會分得頻寬 2.3 Mbits/sec，剩下的頻寬平均分配到兩個 RCST 上，所以 TCP 1 最多取得 2.3 + (7.5 - 2.3) / 2 = 4.9 Mbits/sec，而沒有資料傳送的 RCST 2 分得頻寬 2.6 Mbits/sec，使得頻寬使用率較低。

TCP 2 在第 15 秒開始啓動，TCP 2 的傳輸率上升，而 TCP 1 的傳輸率下降。可以明顯發現，在 FCA-RR 機制下，TCP 1 傳輸率的下降以及 TCP 2 傳輸率的上升很穩定；而在 FCA-PR-0.8 機制下，TCP 1 會因爲部分頻寬轉移至 TCP 2 上，MAC 層可傳送的速率變低，TCP 層產生封包的速率沒有隨著下降，大量封包遺失在 MAC 層，使得 TCP 1 重新進入緩起步階段。以 TCP 傳輸率到達穩定平衡所需要的時間做比較，在 FCA-RR 機制下，兩條 TCP 連線大約 26 秒可達到穩定平衡；而在 FCA-PR-0.8 機制下，需要到 46 秒才開始穩定。

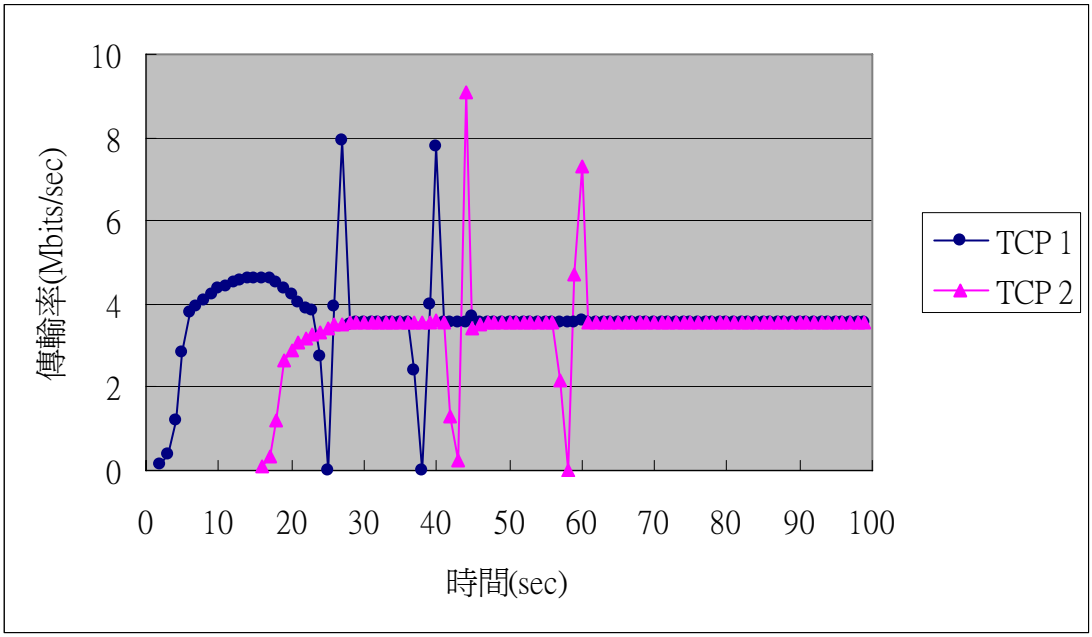


圖 22 FCA-RR 機制下的 TCP 起步

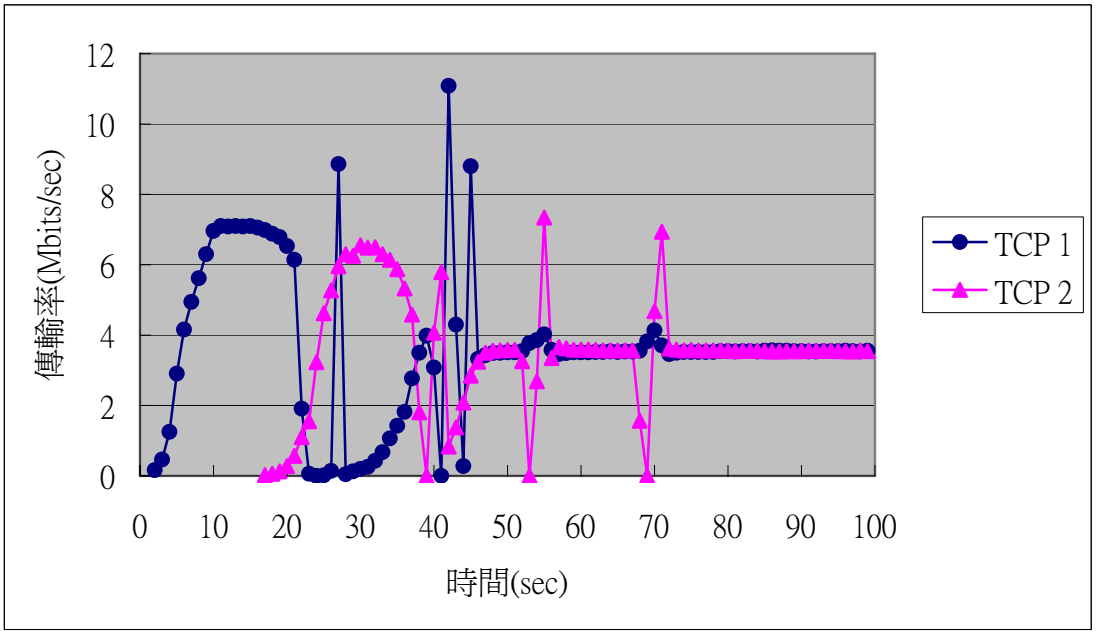


圖 23 FCA-PR-0.8 機制下的 TCP 起步

5.2.1. FCA-PR-m 權數值的影響

在這一節中，我們對 FCA-PR-m 使用不同的權數值做測試，模擬所使用的參數與上一節相同。圖 24、圖 23、以及圖 25 分別是 FCA-PR-0.2、FCA-PR-0.8、以及 FCA-PR-0.95 的模擬結果，我們發現，權數值越小，TCP 傳輸率越慢達到穩定平衡。當 TCP 2 連線啓動時，RCST 2 開始向 NCC 發送 VBDC 要求，頻寬會開始從 TCP 1 轉移至 TCP 2，FCA-PR-m 的權數值 m 會影響到 VBDC 歷史值的變動速率，間接影響到頻寬轉移的速率。當權數值小的時候，RCST 2 的 VBDC 歷史值可以快速上升，所以頻寬快速的轉移到 TCP 2 上，快速的頻寬轉移使得 TCP 1 瞬時產生大量的封包遺失，所以需要較多的時間來回復。

以 TCP 傳輸率到達穩定平衡所需要的時間做比較，雖然 FCA-PR-m 機制在大權數值所需的時間比小權數值還少，FCA-PR-m 機制還是比 FCA-RR 需要的時間還多。

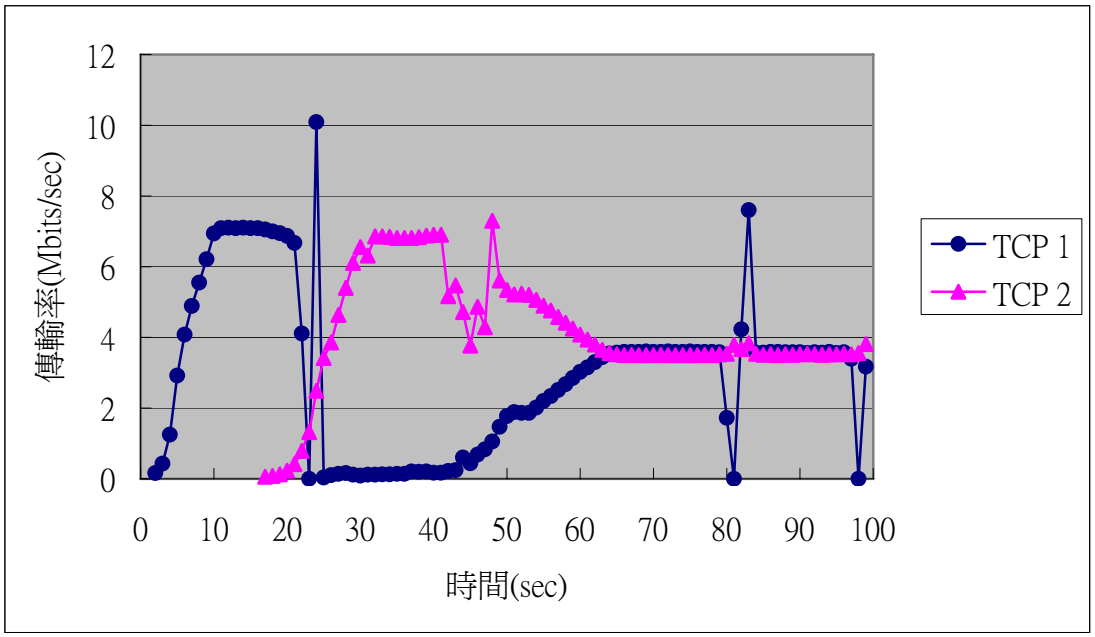


圖 24 FCA-PR-0.2 機制下的 TCP 起步

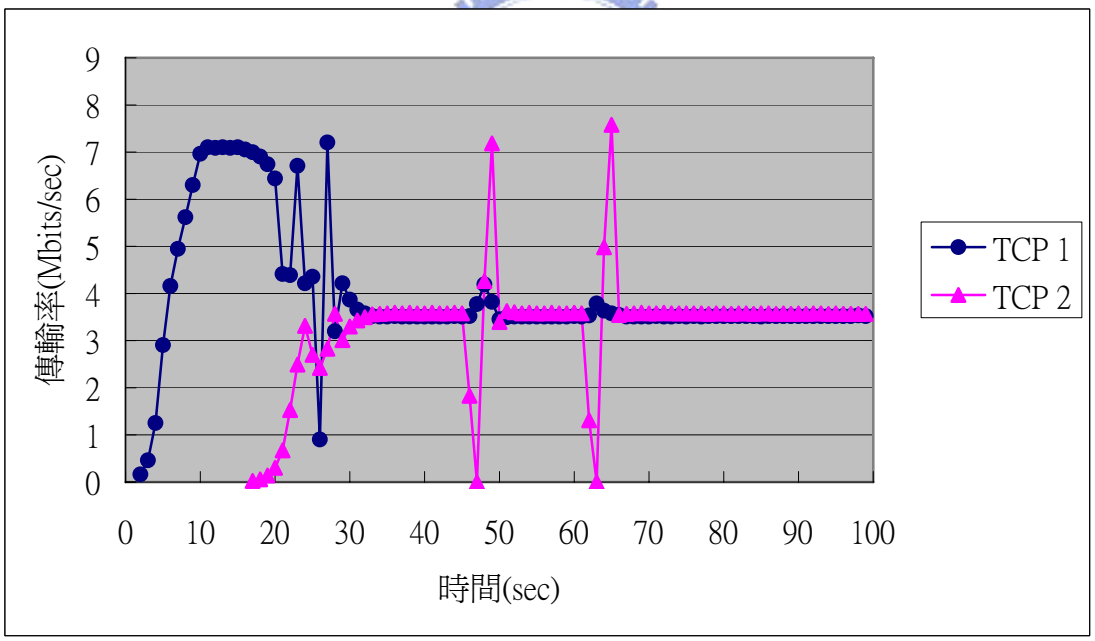


圖 25 FCA-PR-0.95 機制下的 TCP 起步

5.2.2. VBDC_MAX_RATE 的影響

在這一節中，我們調整參數 VBDC_MAX_RATE，觀察此參數對 TCP 效能有何影響。除了 VBDC_MAX_RATE，模擬所使用的參數與上一節相同。從圖 26、圖 27、以

及圖 28 中可發現，對 FCA-RR 機制而言，VBDC_MAX_RATE 的值不影響 TCP 穩定性，有影響的地方在於單獨一個 TCP 連線時，VBDC_MAX_RATE 加大則 TCP 1 可獨得的頻寬也就跟著加大。從圖 29、圖 30、以及圖 31 中可發現，VBDC_MAX_RATE 對 FCA-PR-0.8 的機制影響不大。

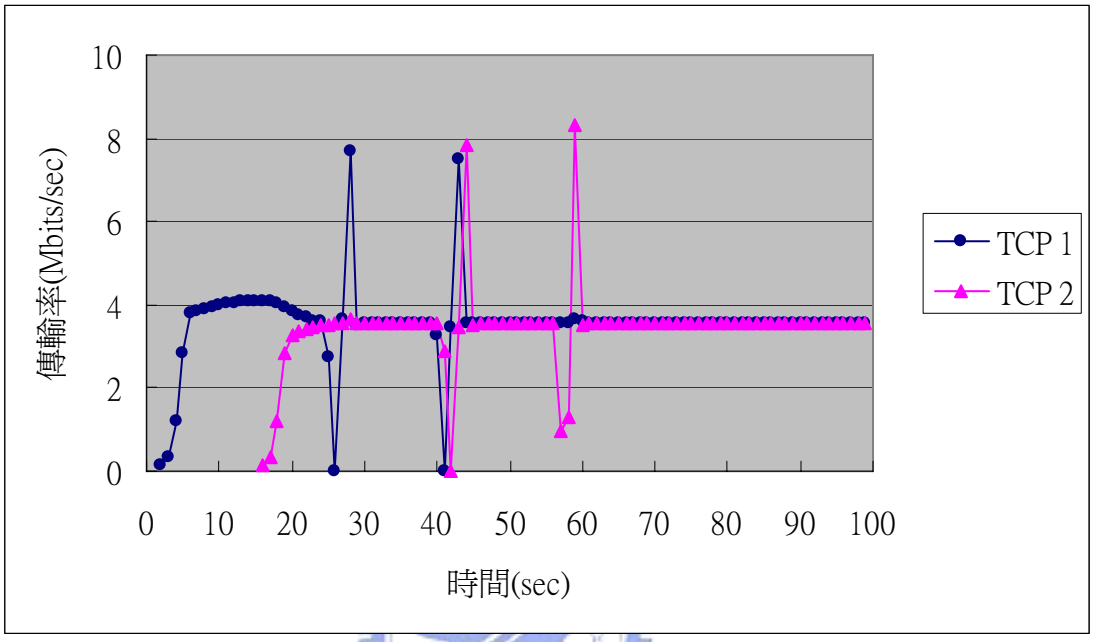


圖 26 模擬結果：FCA-RR 機制搭配 VBDC_MAX_RATE = 1.15Mbits/sec

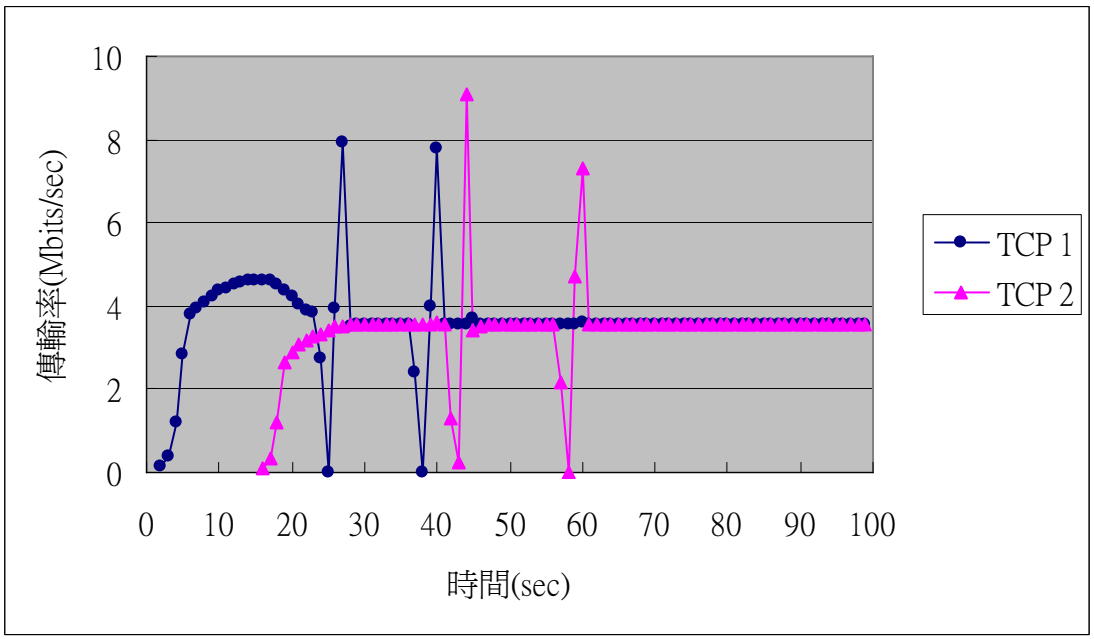


圖 27 模擬結果：FCA-RR 機制搭配 VBDC_MAX_RATE = 2.3Mbits/sec

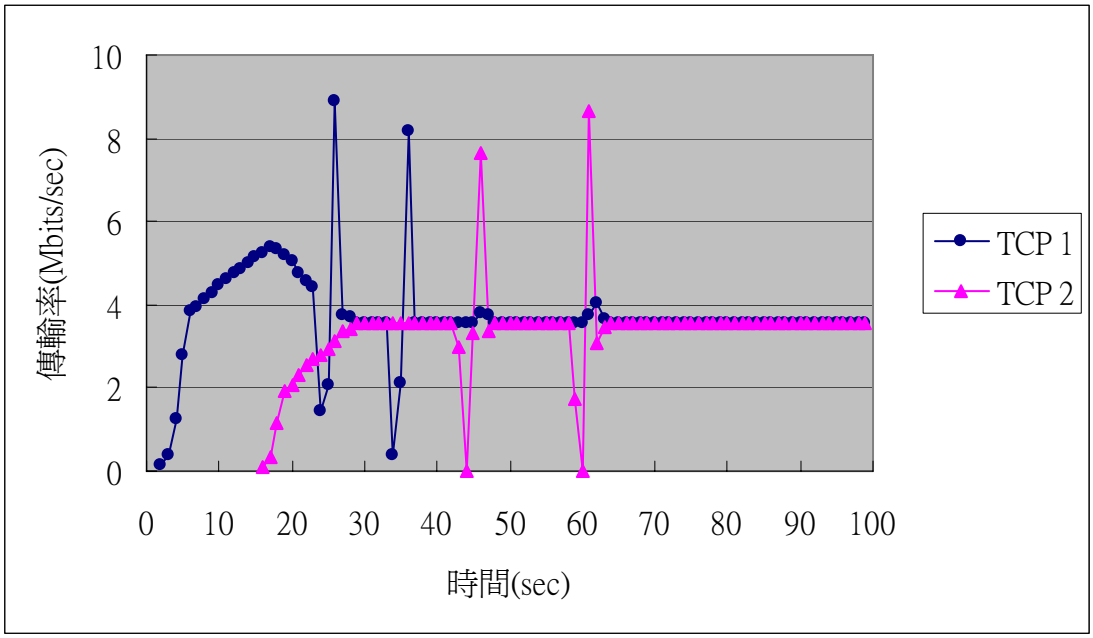


圖 28 模擬結果：FCA-RR 機制搭配 VBDC_MAX_RATE = 4.6Mbits/sec

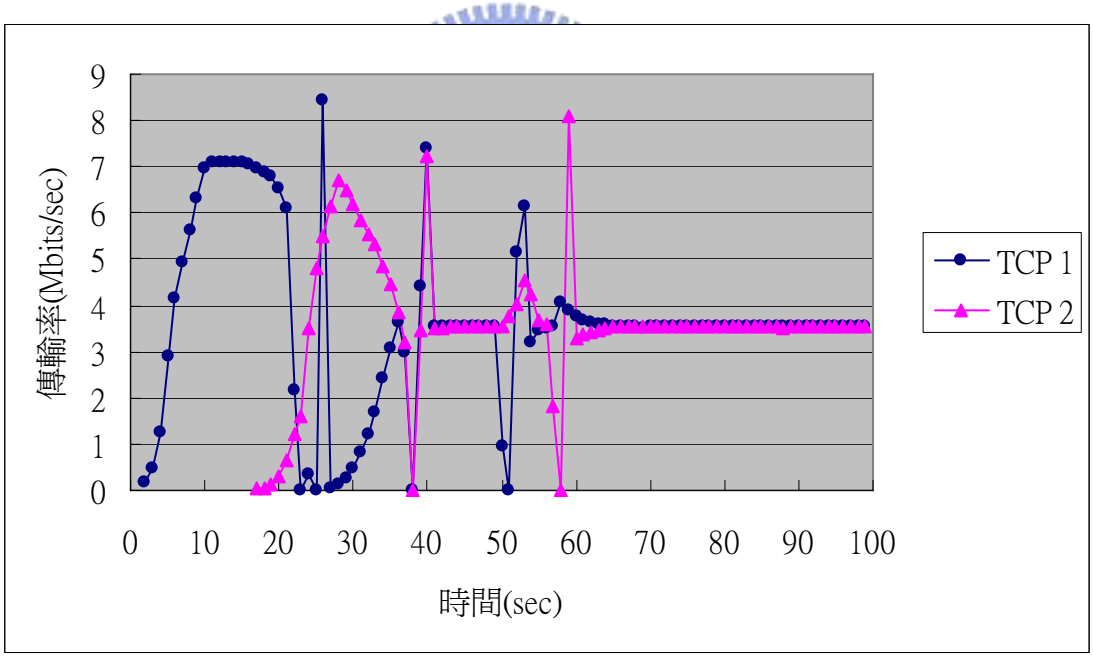


圖 29 模擬結果：FCA-PR-0.8 機制搭配 VBDC_MAX_RATE = 1.15Mbits/sec

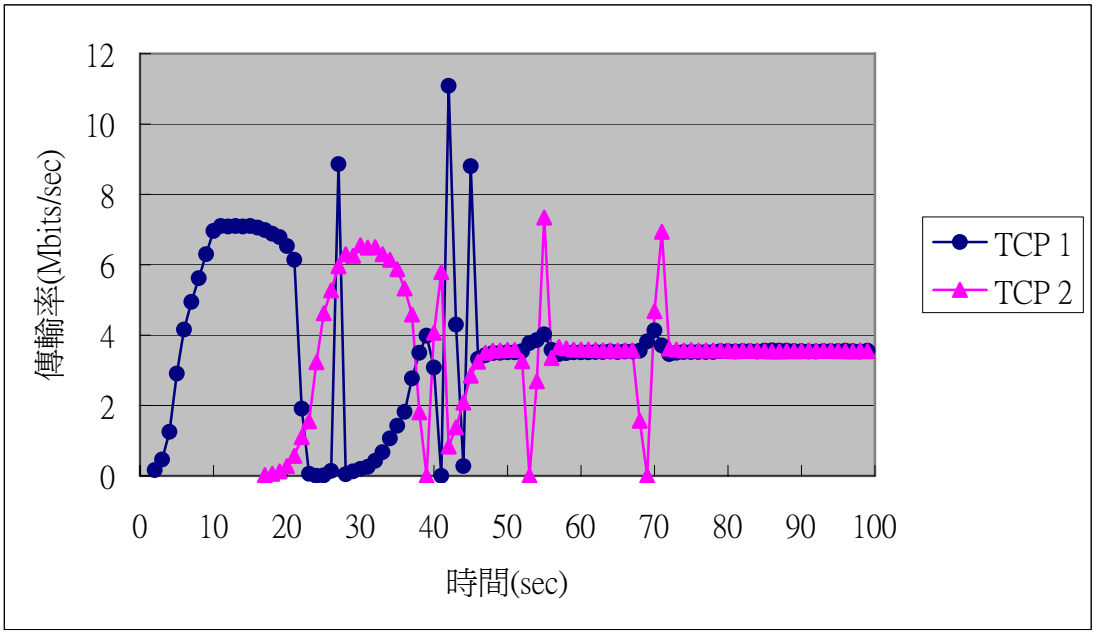


圖 30 模擬結果：FCA-PR-0.8 機制搭配 VBDC_MAX_RATE = 2.3Mbits/sec

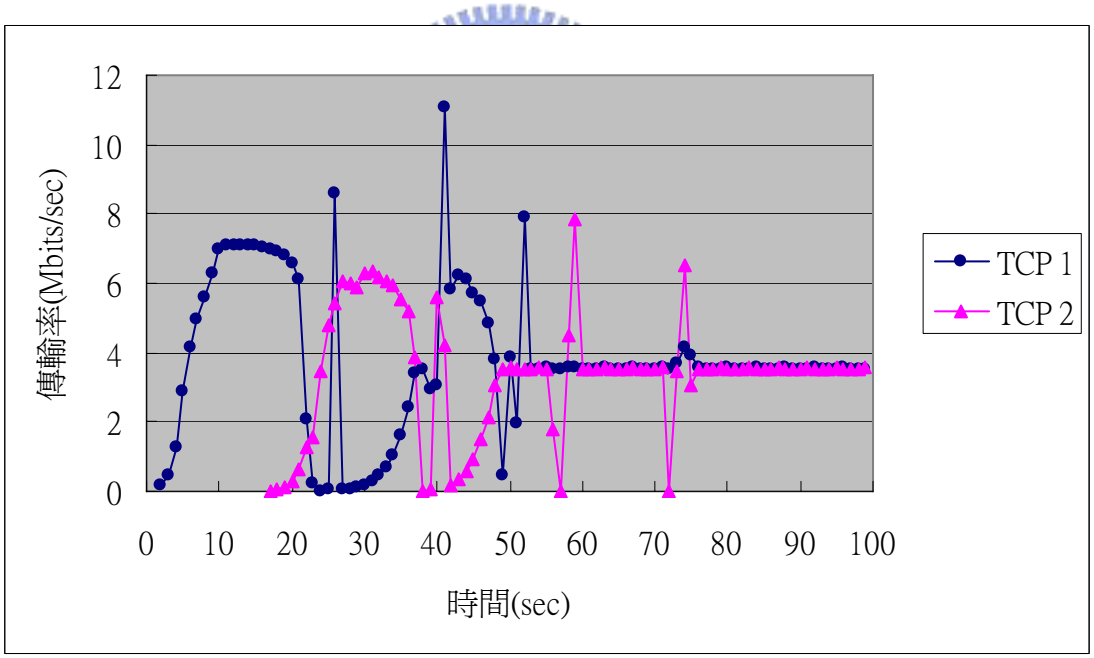


圖 31 模擬結果：FCA-PR-0.8 機制搭配 VBDC_MAX_RATE = 4.6Mbits/sec

六、 結論

衛星網路技術的重要性以及應用性越來越大，許多的研究者探討 DVB-RCS 衛星網路，因此一個高品質而且公開的 DVB-RCS 模擬平台是有價值的。在這篇論文中，我們詳細描述我們在 NCTUns 網路模擬器上所開發的 DVB-RCS 模擬系統，系統中的重要機制皆受到嚴謹的實作、測試、以及驗證。我們實作兩種 FCA 機制，其中的 FCA-RR 以平均方式分配；而另一個 FCA-PR-m 機制按照 VBDC 要求量做分配。藉由模擬，我們觀察到 TCP 連線在 FCA-RR 機制下比在 FCA-PR-m 機制穩定。

參 考 文 獻

1. ETSI EN 301 790 v1.4.1. Digital Video Broadcasting (DVB); Interaction channel for satellite distribution system. September 2005.

2. Pasqiuale Pace, Gianluca Aloi, and Salvatore Marano. Performance Analysis of Connection Admission Control Scheme in a DVB-RCS Satellite System. September 2004.

3. G. X. Chafla Altamiranol, F. J. Ruiz Pifiar, C. Miguel Nieto, and A. Femaindez del Campo. Evaluation of DVB-RCS Resource Assignment Mechanisms for Internet Traffic Transport. September 2004.

4. S. Wang, C. Chou, C. Huang, C. Hwang, Z. Yang, C. Chiou, and C. Lin. The design and implementation of the NCTUns 1.0 network simulator. Computer Networks, 42(2):175–197, June 2003.

5. ETSI EN 301 192 V1.4.1. Digital Video Broadcasting (DVB); DVB specification for data broadcasting. November 2004.

6. ISO/IEC 13818-1. Information technology - Generic coding of moving pictures and associated audio information: Systems. November 1994.

7. ETSI TR 101 211 V1.7.1. Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI). February 2006.

8. ETSI EN 300 468 V1.7.1. Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems. December 2005.

9. ETSI EN 302 307 V1.1.1. Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. March 2005.

附　錄　一

中英對照

適應編碼調製（ACM, Adaptive Coding and Modulation）

代理者（Agent）

分配者（Allocator）

ATM 封包（ATM Cell）

非同步的（Asynchronous）

頻寬（Bandwidth）

動態頻寬機制（BoD, Bandwidth on Demand）

位元組（Byte）

歐洲電子標準化組織（CENELEC, European Committee for Electrotechnical Standardization）

頻道（Channel）

通道編碼（Channel Coding）

通道解編碼（Channel Decoding）

時脈（Clock）

編碼率（Coding Rate）

壅塞控制（Congestion Control）

壅塞視窗（Congestion Window）

控制表格（Control Table）

迴旋碼（Convolution Code）

資料傳送帶（Data Carousels）

資料管道（Data Piping）

資料流（Data Streaming）

延遲（Delay）

延遲差異（Delay Jitter）

反亂碼（De-randomization）

分散式事件模擬方法（Discrete-event Simulation Method）

歐洲廣播聯盟（EBU, European Broadcasting Union）

饋送者（FD, Feeder）

流量控制（Flow Control）

正向通道（Forward Link）

視框（Frame）

頻率寬（Frequency Bandwidth）

保衛時間（Guard Time）

閘道（GW, Gateway）

標頭（Header）

歷史函式（History Function）

交錯器（Interleaver）

聯合專家組（JTC, Joint Technical Committee）

媒體存取控制層（MAC, Media Access Control Layer）

調變（Modulation）

多協定封裝（MPE, Multi-protocol Encapsulation）

MPEG2-TS 封包（MPEG2-TS Packet）

MPEG2-TS 串流（MPEG2-TS Steam）

存取機制（Multiple Access Scheme）

多工／解多工（Multiplexing/De-multiplexing）

網路控制中心（NCC, Network Central Control）

交大網路模擬器（NCTUns Network Simulator）

額外負載（Overhead）

負載（Payload）

協定資料單元（PDU, Protocol Data Unit）

實體層（PHY, Physical Layer）

同步回放（Play Back in Synchronization）

傳輸延遲（Propagation Delay）

偽碼（Pseudo Code）

埠口（Port）

服務品質（QoS, Quality of Service）

佇列（Queue）

亂碼器（Randomizer）

衛星地面接收站（RCST, Return Channel Satellite Terminal）

中繼器（Repeater）

要求訊息（Request Message）

要求者（Requestor）

要求時槽（Request Timeslot）

反向通道（Return Link）

路由（Routing）

里德所羅門碼（RS code）

衛星（SAT, Satellite）

攪亂器（Scrambler）

緩起步（Slow Start）

服務提供商（SP, Service Provider）

超視框（Superframe）

符號（Symbol）

符號傳輸率（Symbol Rate）

同步位元組（Synchronization Byte）

同步化的（Synchronized）

同步的（Synchronous）

時槽（Timeslot）

標尾（Trailer）

資料流（Traffic Flow）

傳輸時間（Transmission Time）

轉送器（Transponder）

通訊資料網路協定（Transporting Data Network Protocol）

資料時槽（TRF Timeslot）

使用率（Utilization）